Title
Contention and Control: U.S. City and Police Responses to the Occupy Campaigns of 2011

Author
Adams, Nicholas B.

Publication Date
2015

Peer reviewed|Thesis/dissertation

Contention and Control: U.S. City and Police Responses to the Occupy Campaigns of 2011

By

Nicholas B. Adams

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Sociology

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Kim Voss, Chair
Professor Trond K. Petersen
Professor Christopher K. Ansell

Summer 2015

Abstract

Contention and Control: U.S. City and Police Responses to the Occupy Campaigns of 2011

by

Nicholas B. Adams

Doctor of Philosophy in Sociology

University of California, Berkeley

Professor Kim Voss, Chair

Research on social movement repression and protest policing has identified four main factors affecting police responses to protest: political context, police capacity, police culture, and the characteristics and actions of the movements police face. However, there has been little consensus about when or under what conditions these factors influence police decisions. Case studies and large-N studies featuring thin data on incomparable cases have not been able to assess the relative strengths of these factors in determining protest policing under varying circumstances. This dissertation treats 184 U.S. Occupy campaigns as a natural experiment on U.S. cities and towns to explore how political context and police factors shape protester and police behavior, and how they do so over the course of protest campaigns. Using innovative text-analysis methods that combine the best of human hand-coding and automated techniques like topic modeling, this dissertation analyses reports of protester and police activities from over 8,000 local, regional, and national news accounts to find (1) that the Occupy campaigns followed a rather similar life course, and (2) that, contrary to going sociological theory, police are influenced by political

elites at the city level and behave strategically. City and police responses are more decisive in cities where political authority is relatively concentrated in an executive. Police are more accommodating of movements when elections are near. Police with relatively small budgets or workforces are more likely to shut down protest campaigns sooner. Departments with relatively fewer officers are also more likely to avoid force-on-force mass arrests, preferring to arrest individuals and smaller groups of protesters. Police departments dedicated to a community policing philosophy are more accommodating of movements, and more likely to focus enforcement efforts on individuals rather than engage in group punishment. And, departments in cities with high violent crime rates are more likely to take a nonchalant approach to protest campaigns.

Table of Contents

Acknowledgements

The research work resulting in this dissertation was a team effort. I am forever grateful for the thousands of hours of hand-coding and data collection performed by dedicated, brilliant, and good-hearted research assistants including: Carly Vendeiro, Sofie Garden, Marybeth Baluyot, Christiana Dunlap, Emily Cramer, Lauren Meyer, Gladys Aparicio, Jonathan Scott, Elena Nielsen, Elaine Kliner, Ari Hosseini, Jenny Segura, Elliot Spector, Ariana Braga, Roxie Bostwick, and Tina Tseng.

When I began this project, I was neither a computer programmer nor a data scientist. A number of people have helped me become at least the latter, patiently contributing their consulting expertise and/or code to this project. Aaron Culich has been a close and dedicated collaborator on the aspects of the data preparation that required coding in java to work with Stanford's CoreNLP. He volunteered many late-nights to make this dissertation possible. Steven Rivera volunteered his time early in the project to write the code translating our team's hand-coding into a machine-readable XML format. Many of my colleagues at the Berkeley Institute for Data Science (BIDS) also pitched in to help me through programming snags in python and R. Special thanks go to Stefan Van der Walt for his generous spirit and quick work in python; to Lauren Ponisio for her enthusiastic help with the data plots I created in R; and to Karthik Ram for his gracious help coding the functions to identify dates in text, and processing earlier data. These three people and others at BIDS – including Daniel Turek, Dani Ushizima, and Katy Huff (who each helped me through other smaller snags in R) – offered technical aid and encouragement whenever I needed it, without fail. Together, they helped me maintain a buoyancy of spirit through the difficulties of the computational work, transforming an arduous, anxiety-inducing process into something that was, I dare say, enjoyable. I feel extremely lucky to have been surrounded by all these talented and giving colleagues. I am, therefore, thankful as well to Saul Perlmutter, Cathryn Carson, Henry Brady, Josh Bloom, Kevin Koy, Fernando Perez, Henry Brady, Josh Greenberg, Chris Mentzel, the Alfred P. Sloan Foundation, the Gordon and Betty Moore Foundation, and everyone else who has made the Berkeley Institute for Data Science what it is.

I also want to thank the D-Lab. I have been very fortunate to enjoy the support of my fellow staff members there, and to learn so much from the young scholars who make it go. D-Lab

hosted the Text Analysis workshop by Laura Nelson that got me moving in this direction. And it was a tip from D-Lab consultant Chris Krogsland that pointed me toward the structural topic modeling package that this dissertation deploys. Leading the Computational Text Analysis Working Group at D-Lab has also propelled my work forward. It is a privilege to work alongside so many bright minds while wrestling with new methods and tools. I also wish to thank the National Science Foundation, which supported this dissertation with an early Doctoral Dissertation Improvement Grant (#1303662). And I would be remiss if I did not thank the many campus staff and colleagues who have contributed to the forward momentum of this project in ways large and small. Thanks, in particular, go to Anne Meyers, Carolyn Clark, Ali Ferguson, Stacey Dorton, Susan Grand, Jon Stiles, Patty Frontiera, and Claudia von Vacano.

It is very important to me to thank the handful of people who have supported this work from the beginning. It is rather uncommon for a project of this magnitude to be undertaken by a doctoral student. It would not have gotten off the ground but for the early encouragement of Kim Voss, Claude Fischer, Trond Petersen, Chris Ansell, and Cathryn Carson. Votes of confidence from Laura Nelson (who also encouraged me to try out automated text analysis!), Jen Schradie, Pablo Gaston, Brian Lande, and Rachel Pritzker also propelled me forward. I also thank Kim Voss, Trond Petersen, and Chris Ansell, in particular, for their valuable feedback on the dissertation, notes that have aided the writing in its current form and will improve its future products.

For moral and professional support and mentorship, I also wish to single out Cathryn Carson for special appreciation. When this project was nearly foundering for lack of financial and technical support, she helped bring me into the Berkeley data science community. And at various other points of temporary but seemingly catastrophic existential crisis, she kept my head above water. The project would not have succeeded without her. Family and close friends have offered me moral support throughout this process as well. I wish to thank Elaine, Mark, Laura, Patrick, Margarita, Tony, Lindsay, Ben, Jan, Helena, Benjamin, CMoore, Taylor, and Tyler for keeping me healthy and cared for.

I am so grateful for Kim Voss. I needed precisely the person, scholar, and mentor who is Kim Voss to have pulled all of this together. She introduced me to the social movements literature. She sponsored my apprenticing of research assistants. She encouraged and believed in what I was doing from the start, which allowed me to feel like it was, indeed, doable. And all the while, she catalyzed my growth as a scholar through her dissertation

group. (That group has included the powerful minds and good hearts of Laura Nelson, Pablo Gaston, Jen Schradie, and Kristin George.) There, Kim has modeled her mastery of not only providing useful and incisive suggestions, but doing so with care and constructive intentions. And, it turns out, I needed this caring, constructive, rigorous encouragement most of all. Thank you, Kim.

Finally, I must end where I began with the team of research assistants who put their minds, hearts, and labor into this work. They are wonderful people – bright, responsible, honest, dedicated to rigor, and driven to make a positive difference in the world through good work, however painstaking. We made each other better. We lifted one another's spirits, carrying each other through drudgework and difficult personal times, alike. We faced daunting technical challenges and we bested them. We laughed. We groaned. (It was funny!) And, sometimes, we cried. … At the risk of sounding ridiculous (to paraphrase another change-seeker), this project has been motivated by a love for humanity and a hope for a functioning democracy of ideas. While I am pleased that it has produced useful scholarship after so much toil, I also feel nostalgia for the process we shared in together. Thank you to all. We did it.

Chapter 1: Introduction

In the last few years, a wave of protest has swept the world: The Arab Spring, the Occupy movement, anti-austerity rioting in England, police and protester clashes in Spain and Greece, labor unrest in China, uprising across Ukraine, mass marches in Thailand, demonstrations against racist police brutality in the United States—the list goes on. The outcomes and storylines of these protest movements depend in large part on the popularity of the reforms they seek, the sympathies of the governments they target, and the allies they can attract to their cause. But they also depend greatly on the ways protest campaigns are policed.

This last claim should surprise no one. Yet, despite decades of research on contentious politics describing thousands of contentious events and scores of campaigns, there is strikingly little settled theory reliably explaining protest policing and its dynamic relationship to protest.

Scholars are not failing outright. And they are not disagreeing to be disagreeable. The cases they study are simply different. While scholars have mostly agreed that protester and police interactions are shaped by political context, governing capacities, police culture, and the claims, strategies, and tactics of challengers, they have not been able to gain clarity about the conditions under which these factors matter more or less, how they interact, or why each set of factors seems to matter so much at some times and not at others. Achieving that level of detailed understanding – one that teases out the dynamics of the interacting factors influencing the unfolding of protest campaigns and their policing – requires numerous comparable cases of very similar contentious campaigns. The Occupy movement, with its 184 campaigns across the U.S., presents a rare opportunity to study questions plaguing researchers for decades.

To give a sense of the challenges faced by researchers trying to understand governments' interactions with contentious movements when those movements are not comparable, consider a question that has long been relevant to researchers, movement strategists, and

authorities facing protest campaigns: *Does government repression of a protest march scare protesters away from future participation or rally them to the cause?*

Carey's (2006) studies of South American and African protest movements offer a clear answer: repression tends to chill movement support. In contrast, Opp and Roehl's (1990) research describing the effects of repression on anti-nuclear protesters in West Germany or Moore's (1998) analysis of Peru and Sri Lanka, however, concludes that repression causes a movement to rally. Then again, Karstedt-Henke (1980) offers evidence that repression may cause movements to split into separate factions pursuing radical and institutional strategies. Some scholars have even argued convincingly that repression may chill, *and then* rally, a movement (Rasler 1996) – a reverse of the phenomenon which may be familiar to observers of Occupy Oakland.

As Christian Davenport, an elder among political scientists studying contention and repression laments:

> "Repression has been found to have every single influence on behavioral challenges [protest], including no influence… Results have shown that repression increases conflict, decreases conflict up to a certain level of repression and then increases it—a U-shape, decreases some forms of dissent while increasing others, and has no impact whatsoever. As a result[…], we know very little about how repressive behavior influences [protest]..." (Davenport and Inman 2012, 624).

Contradictory as these accounts are, they are all likely correct so far as they go. They correctly describe the phenomena researchers observed for the cases they studied. To overcome the frustration voiced by Davenport, scholars must pose a question much more complicated than the simple one italicized above: *<u>Under what conditions and at what points in the life course of a campaign</u> does government repression of a movement scare protesters away from future participation and when does it rally them to the cause?*

Answering such a question is very difficult. It requires that one account for multiple factors influencing an outcome, including time, *and* that one collect data on a relatively large

number of comparable campaigns to tease out the relative effects of those factors. Until the Occupy movement, this simply was not possible.

The lack of comparable movements has so hindered the study of protest policing, in fact, that not only have the above questions gone unanswered; many, much more more basic, questions have also gone unanswered. The social scientific literature on protest policing is still unclear, for instance, about whether (and under what circumstances) police responses to protesters are shaped by political elites. There is little consensus, either, about how very basic features of a police department – like its budget and number of officers – affect its response to protest. And virtually no comparative work has assessed the extent to which a police department's culture – its philosophy about its role in the community and view of what it means to be police – shapes its interactions with protest movements.

Instead, scholars have pursued one of two programs. They have contributed single cases studies to a slowly accreting literature identifying actions, mechanisms, processes, and structural factors *likely* to affect the interactions of, and outcomes for, contentious movements and the governments that respond to them. Or, scholars have collected thin data on many, many cases of political contention that are not particularly comparable. Chapter 2 describes the victories and defeats of these research programs in more detail. The first program offers thick data useful for understanding the range of phenomena and factors that should interest researchers, but lacks enough comparable cases to tease out how those factors interact to produce particular outcomes. The second program captures very thin data on many cases that are so different they cannot easily be compared to produce valid findings.

Overall, this research on protest policing has identified four main factors likely to determine how police manage protest: the governing context under which police and protesters operate, police departments' capacities, police departments' cultures, and the characteristics and activities of the protest movements they face. No study yet, however, has been able to assess the relative strength of these factors in the determination of protest policing over the life course of a social movement *campaign*, the subject of this dissertation.

The study of protest movement campaigns – sequences of protest events all driven by (substantially) the same goals, claims, and (strategic) actors – will allow researchers to better understand not only how city, police, and movement factors affect protest policing

through time, but also to uncover evidence that police and protesters act strategically. So far, quantitative researchers attempting to understand police responses to protest have only compared cases of (disparate) protest events as one-off phenomena. For instance, analysts have treated the second march in a series of protest events seeking equal rights for women as though it were attended by entirely different people beginning a protest movement anew. Such a conceptual mistreatment of protest campaigns forecloses analyses revealing strategic planning and interaction between movements and the police charged with controlling them. As a consequence, some of these researchers have mistakenly concluded that police do not act strategically when facing protest movements.

Key Questions

This dissertation endeavors to answer the following questions about protest policing:

> *Q1: What do police do?: How do police respond to protest in the contemporary United States? What sorts of operations and tactics do they typically use?*
>
> *Q2: Why do police do what they do?: Do the broad factors thought to influence protest policing – features of the governing context, police capacities, police cultures, and movement variations – indeed influence police responses to protest?*
>
> *Q3: Do performances vary over campaigns?: If these broad factors do influence policing of protest, do their effects vary over the course of protest campaigns?*
>
> *Q4: Do police act strategically, or do they simply react to protesters?*

Questions 2, 3, and 4 can only be answered statistically if the variation from Occupy campaign to Occupy campaign is rather limited. Therefore, this dissertation must also ask:

> *Q5: Were Occupy campaigns similar? Did they engage in similar contentious performances (i.e. Marches, Rallies, Demonstrations, Occupations) from city to city? To the extent that prevalence of performances varied, can that variation be assessed and controlled for by measurable covariates?*

Even, if the answers to Question set 5 are affirmative, Questions 3 and 4 cannot be answered unless Occupy campaigns also proceeded according to a rather similar timeline.

Therefore, the dissertation also investigates the extent to which Occupy campaigns were similar and comparable when it comes to the sequence and timing of their activities.

> *Q6: Did Occupy campaigns proceed according to a common life course, engaging in a common sequence of activities with similar timing?*

Research Design

This dissertation overcomes the limitations of previous research attempting to understand police and protester interactions by treating the 184 U.S. Occupy campaigns as if they were a 'natural experiment' on U.S. cities. Readers might imagine a team of scientists creating 200 identical movements (pretending for the moment that such a thing is possible) and placing each one in a separate U.S. city or town. In this hypothetical reality, scientists would be able to conclude that any differences they observed in the actions of protesters, police, or government officials, from city to city, resulted from differences among the cities.

Occupy, of course, was not created in a lab of scientists. And each campaign of the movement was a bit different than its siblings. Some were larger and more active than others and they were composed of different individuals. But the U.S. Occupy campaigns are very similar compared to the range of protests often compared side-by-side in large-N studies of contentious politics and protest policing. The differences between Occupy movements, too, are measurable in ways that allow for the statistical control of those differences.

This dissertation project began with the collection of over 8,000 news accounts – every local, regional, and national newspaper, radio, and television account between September 1st and December 30th 2011 – describing Occupy-related activities of protesters, police, or city government officials. Via a new text analysis approach combining the strengths of human hand-coders and automated algorithms, this project identified Occupy events of interest and extracted information about the activities occurring within them. This text analysis pipeline is described in detail in Chapter 3 and includes methods designed to extract information about actors and actions across cities, then normalize those entities so that similar phenomena may be counted and compared statistically in light of variables describing city contexts and police department capacities and cultures. (These methods

include algorithms performing co-reference resolution; custom actor dictionaries to identify government officials, police, and protesters; a clause-based subject-verb-object triplet extractor; and structural topic modeling of SVO-amplified text units through the duration of Occupy campaigns – all explained in Chapter 3.)

This dissertation's approach inductively identifies a range of *contentious performances* used by protesters and a range of *control performances* used by police. Analyses measure the prevalence of these performances at different points in time and in light of variables thought to affect protester, police, and government behaviors. Thereby, this dissertation tests many hypotheses about the circumstances under which this or that *performance* is enacted by protesters or police, ultimately answering Questions 2 and 3 above.

Overview

This dissertation continues, in Chapter 2, with an introduction to literatures on social movements, protest policing, and social movement repression. The chapter identifies a number of apparently conflicting findings from sociologists and political scientists, and three broad hypotheses about protest policing motives.

Chapter 3 presents a detailed account of the data, methods, and research design brought to bear on the key questions introduced above. It explains the text processing pipeline that renders data suitable for testing the hypotheses identified in Chapter 2 and further specified in Chapters 4, 5, and 6.

With a clear understanding of what is known, what questions still require answers, and how the methods and data of this dissertation are able to provide those answers, Chapter 4 moves into discovery and analysis. Because all the later chapters concerning protest policing activities depend on an understanding of the protest activities to which police and cities are responding, Chapter 4 begins with a thick quantitative description of Occupy campaigns' activities. This chapter not only shows how Occupy campaign activities were constrained in a coherent, countable set of contentious performances, it begins to show that the timing and sequences of those performances were not random, but, in fact, conformed to a common campaign life course with only minimal (and measurable) variation. This deeper understanding of a constrained set of variables describing protester

activities will enable clearer understandings, in later chapters, of what police and cities were facing when they made decisions about how to interact with Occupy campaigns.

But just as city and police behaviors do not occur in a vacuum, neither do protester activities. A mature body of social movements theory, discussed in Chapter 2, predicts that protester activities will be shaped by the political context, the 'political opportunity structures,' in which they emerge. This theory, developed initially by Doug McAdam, Sidney Tarrow, and Charles Tilly (McAdam, Tarrow, Tilly, 2010: McAdam 2010; Tarrow 2013), has helped to explain differences across contentious campaigns from decade to decade and under various national regimes. But few have observed the effects of political opportunity structure at the scale of days and cities (though see Amenta and Zylan 1991 for a study of U.S. states).

Chapter 5 extends previous theory to the high-resolution scale of cities and days. Do small differences in political opportunity structures – the sorts of variation in government type, electoral insecurity, and political ideology measurable across U.S. cities – affect protester performances? Do these factors influence protester activities more at some points in their campaigns than at others?

With a baseline understanding of what protesters are up to and how political opportunity structures impact their activities, Chapter 6, turns to an exploration of police control performances. The review of protest policing literatures in Chapter 2 highlights a number of hypotheses that have been rather difficult to test prior to the Occupy movements' provision of so much comparable data. Chapter 6 begins by applying the same techniques used to understand the range and prevalence of protesters' contentious performances to textual data describing police-initiated events during the Occupy movement. Next, Chapter 6 tests a number of hypotheses suggesting that police departments' capacities and cultures influence the way they police protests.

The findings of Chapters 4, 5, and 6 are reviewed in Chapter 7. Their implications are discussed. Conclusions highlight the extent to which police departments appear to determine their own actions, a corrective to some researchers' (Earle and Soule 2006) suggestions that police responses are mostly driven by protesters' behaviors.

Theoretical Contributions

The limitations of current repression and protest policing theories result primarily from the complexity of the phenomena they seek to explain. It is very difficult to compare the policing of one protest movement to the policing of another which features different aims, strategies, and/or tactics. That difficulty is compounded if the political contexts in which protest policing occurs differ considerably. With so much variability, the conclusions of case studies and small-N comparative studies often contradict one another, and prove limited in generalizability.

After years of research developing limited and tenuous conclusions, scholars in this subfield have yearned for increased comparative leverage (Koopmans 2004). But, it is incredibly rare for any movement contentious enough to draw a protest policing response to also spawn independent campaigns enacting similar performances across a large number of contextually-similar cities in a short period of time. Fortunately, the Occupy movement of 2011 did just that. This dissertation capitalizes on the rare historical circumstance of the Occupy movement by treating it as a 'natural experiment' enabling the investigation of a number of questions still unanswered by scholars of protest policing.

This dissertation makes three significant theoretical contributions to three literatures.

1. It clarifies for the protest policing and repression literatures when and how much political opportunities, police departmental capacity, and police departmental culture affect the policing of protest movements (occupation campaigns, in particular).
2. It clarifies for the American sociological literature on protest repression the extent to which police responses to protest movements are reactive, strategic, and or motivated by their sense of threat to their control.
3. It demonstrates for the contentious politics/social movements literature the value of studying *performances* within *campaigns*, particularly the ability of such studies to reveal strategic learning and thinking on the part of actors.

Chapter 2: Literature Review and Theoretical Motivation

The key questions of this dissertation concern how US police respond to protest in the contemporary United States, why they respond as they do, and how and why those responses vary over the course of movement campaigns. Political scientists have sought answers to broad questions about governments' response to challenges since the discipline began. In recent decades, they have also developed a *Protest Policing* literature focusing more closely on what happens when contentious groups meet police in the streets and public spaces. Sociologists have joined this discussion more recently, mostly as an extension of a broader literature seeking to explain social movement phenomena. But, sociologists' focus also tends to merge with longer-standing sociological preoccupations with power struggle and legitimation of authority (in the Marxist and Weberian traditions). Hence, the sociological literature on protest policing has been couched in a broader discussion of government *Repression* of social movements and questions about the extent to which police enact the will of elites or behave with relative autonomy.

This dissertation bears on and draws from all of these literatures to investigate four major factors affecting police responses to protest: city political structures, police capacities, police cultures, and protesters' behaviors. This chapter reviews literatures discussing, in turn, what police do when they face protest, how these four factors influence police responses to protest, and how this dissertation improves upon existing theory seeking to explain police behaviors. While conclusions about the effects of these factors on protest policing are rather discordant – reflecting the broad variation of cases across small-N studies inadequate for comparison – the going theory developed by sociologists studying protest policing quantitatively suggests that police behavior is largely autonomous from political elites and reactive to protesters' on-the-ground actions.

Some limitations of the studies generating this theory of reactive and threatened protest policing point to the need for this dissertation's research, which tests three competing theories explaining the motivations behind protest policing: that police behave in *reaction* to protesters, that they act *strategically*, and/or that they respond based upon the level of *threat* they perceive. Finally, since testing whether police behave strategically requires measuring their behaviors in a temporal context able to discover planning and responses across the duration of a protest campaign,, this chapter reviews social movements literature conceptualizing the time-dimension of protester and police activities.

What do police do?

The foundational task of describing and cataloguing protest policing styles, strategies, and tactical repertoires is ongoing. (Readers might see della Porta and Reiter 1998 for an early review; Davenport and Inman (2012) for a political science review; and Earl (2011) for the latest sociological review) Throughout Western Democracies, authors have noted a shift toward softer and more accommodating protest policing tactics from the late 1960s through the 1990s. (See Waddington (1994) on the British case and della Porta and Reiter (1998); Winter (1998); Fillieule & Jobard (1998); and Jaime-Jimenez and Reinares (1998) for continental cases.)

In the US context, McPhail, Schweingruber, and McCarthy (

1998) identified two main protest policing approaches. The first, 'escalated force,' was commonly deployed in the 1960s and early '70s, and was characterized by police disregard for free speech and assembly rights, intolerance for disruption of normal traffic or routines, disinterest in communication or negotiations with protesters, indiscriminate arrest of protesters, and dramatic shows of force quickly escalating to the use of batons, tear gas, electrical cattle prods, and dogs. The authors describe how this protest policing approach, was replaced in the 1970s by a 'negotiated management' approach characterized by the protection of free speech and assembly rights, the tolerance of "an acceptable level of disruption," extensive interaction with protesters to plan for safe protest activities, a minimal use of arrests with a preference for keeping the peace over strict law enforcement, and the "minimum necessary use of force" when protecting persons or property and arresting lawbreakers (Ibid.).

More recently, authors have argued that protest policing is still evolving in the United States. Noakes and Gillham (2007) concluded, after observing police tactics used to manage

the 1999 WTO protests in Seattle, that police now use an approach centered on the 'strategic incapacitation' of movements. Police remove protesters from their targets by shunting them into protest zones; they use "less-lethal" weapons that are, in fact, rather injurious; and they arrest protesters strategically, removing them from the field of action. The method of 'strategic incapacitation' often also includes covert operations like surveillance and movement infiltration. Vitale (2007) argues that he has also identified a "command and control" model and a "Miami model" based on observations of New York Police Department and Miami Police Department protest policing styles, respectively. Waddington (1994), for his part, has coined the terms 'soft hat' and 'hard hat' protest policing to capture the differences between approaches relying primarily on the limiting of protests through extensive pre-event negotiations and those that use more overt coercion and force.

Even more recently, researchers,say they have identified a new style of protest policing that responds, in particular, to protests around international summit meetings. Since the 1999 Battle in Seattle, argue Starr, Fernandez, and Scholl (2011) a network of security experts and consultants has grown up to help international cities defend against "alterglobalization" protests seeking to disrupt meetings of the IMF, G8, WTO, and World Bank. Host cities largely follow the lead of these consultants, deploying their own officers and private security personnel to pre-emptively criminalize dissent and assembly, remove protesters from the scenes of meetings, intimidate protesters with the use of less lethal weapons, and, in many cases, jail them until meetings are completed. Starr et al (2011) stress that this evolving style of 'strategic incapacitation' merges physical and psychological force to weaken, frustrate, divide, and discourage protesters.

Others have been pointing up the gradual 'militarization' of protest policing since the 1960s. Kraska and Paulsen (1997) have argued that SWAT teams have been used as the "iron fist" inside the "velvet glove" of negotiated management approaches for decades. This debate (including Kraska and Kappeler 1997; Reiner 1998: Soule and Davenport 2009; Waddington 1999) continues, but it is telling that McPhail and McCarthy (2005), the first to highlight the trend from escalated force to negotiated management protest policing, have acknowledged that protest policing appears to rely, increasingly, on the use (or threat) of paramilitary police force. Della Porta (2013) has begun to focus more energy here as well, arguing that protest policing has become more aggressive.

Authors like Starr and Fernandez (2009) have attempted to link this uptick in state repression to shifting state security priorities in the aftermath of 9/11. Cunningham (2004), too, has argued that many of the domestic surveillance practices outlawed after the exposure of COINTELPRO were re-instated after 9/11, and that these may negatively impact movements. Though the impact of these often covert activities are difficult to assess, it does seem probable that counterterrorism and counterinsurgency tools and methods diffuse from militaries to police departments. Researches have found, for instance, links between Reagan's counterterrorism initiatives and governments' repression of movements (Zwerman 1989).

The identification and naming of broad protest policing styles can provide a useful short-hand for scholars, but this common practice becomes problematic if it leads researchers to believe that a country, state, or even individual police department always (or even primarily) uses a signature approach whenever it faces a protest movement. Earl , Soule, and McCarthy (2003), in a (rare) large-N comparative study of protest policing in NY State during the 1960s and '70s, eschewed this trend. Instead of seeking to identify overall approaches or styles, the authors conceptualized protest policing in terms of smaller sets of tactics that logically fit together. They found that police deployed an array of their hypothesized tactical repertoires at various protest events, including doing nothing; observing from a distance; setting up barricades to limit protesters and only making selective arrests; using extensive force and mass arrest; and using a combination of many of these approaches in a single event.

Fortunately, empirical researchers have continued to clarify the literature's focus on the range of tactics that police use (rather than just identifying trends in broad approaches). This dissertation will follow and extend this approach, collecting data on an array of police activities, then examining them to determine what factors influence the policing of protest.

Measuring Tactics

There are a number of ways authors have sought to conceptualize and operationalize specific protest policing activities. Early on, Gary Marx (1979) began the work of constructing lists of policing tactics, work carried on by Carley (1997). Some authors have built indices around such lists (Davenport 2007), or organized their elements along continua from most to least "repressive" (McPhail and McCarthy 2005). Such operationalizations have enabled large quantitative studies seeking, since Feierabend and

Feierabend (1966), to quantify repression across national contexts. Thus, Brockett (1995), Nepstad & Bob (2006) Almeida (2008), Ondetti (2006), and Loveman (1998) have studied levels of repression across Central and South America. Alimi (2009), Alimi and Meyer (2011), and Rasler (1996) have focused on the Near East. Johnston (2006) has focused on former Soviet States. Chang (2008) and Zwerman and Steinhoff (2005) have focused on Asian and Pacific Island countries. And others, still, have compared across developing countries (Davenport 1994) or non-democratic states (Osa and Schock 2007).

These large-N studies (e.g., della Porta and Reiter 1998) typically seek to explain trends in protest policing styles over decades, or their geographic distribution across nations-states. They are not designed to explain why police in a particular place or time chose to fire tear gas on a particular crowd as opposed to accommodating their protest march.

Researchers performing smaller-N, qualitative studies, on the other hand, have spent more energy operationalizing protest repression tactics and developing theories to explain their use. Thus, Boykoff (2007), identifies mechanisms of repression like 'stigmatization,' 'intimidation,' and 'resource depletion." Oliver (2008) categorizes repressive tactics by the goals apparently motivating them: 'incapacitation,' 'deterrence,' and 'surveillance.'

Earl (2003) offers a useful 12-type taxonomy that appears to successfully categorize *all* types of repression. Each repressive tactic, operation, or strategy may be distinguished by the following three criteria:

> "(a) whether the repressive actor is a state, private, or hybrid actor; (b) whether the repressive action is coercive or uses more carrot-based "channeling" (McCarthy, Britt & Wolfson 1991, Oberschall 1973); and (c) whether the repressive action is observable/overt or unobserved/covert." (Earl 2011, 264).

The police activities investigated in this dissertation are most likely to fit into Earl's first type for all three criteria. They are actions taken by states (a1), typically coercive (b1), and overt (c1). More specifically, they are actions taken by city police with the goals of managing or controlling local protest movements. As such, this study fits squarely in the broader sociological literature on repression, and the narrower political science literature on protest policing.

The method this dissertation uses to identify and operationalize police activity is new to the field and explained fully in Chapter 3. For now, readers may appreciate that it departs from previous work in one significant respect: it begins with induction. Rather than searching through news reports by hand for evidence of researcher-defined actions, the method allows relevant police activity to emerge through a text analysis technique able to measure police (and protester) activity in large quantities of news data. While this inductive approach does not align with previous researchers' operationalizations of policing tactics – This dissertation cannot directly count and measure the occurrence of policing tactics studied by other researchers. – it does recover ample evidence of those tactics. As we will see in Chapter 6, police responding to local Occupy campaigns used a broad range of tactics and approaches commonly identified in the literature. This dissertation focus most attention, however, on understanding *why* police do what they do.

Why Do Police Do What They Do?

With a decent understanding of *what* police do, the sociological and political science literatures on repression and protest policing have spent the last decade or so trying to explain *why* they do what they do.

Both literatures agree, broadly, that three main factors are likely to determine how police manage protest: *the political and governing context* under which police operate (McCarthy et al. 1991; Davenport 1997; Wisler and Kriesi 1998; Della Porta 2006), *police departments' capacities and cultures* (Della Porta 1998; Winter 1998; Earl et al. 2003; Worden 1989), and the *characteristics and activities of the movements* they face (Soule and Davenport 2009; Earl et al. 2003; Earl and Soule 2006; Davenport and Soule 2011). But to date, no consensus has emerged to explain which of these factors matter most in determining protest policing, nor under what circumstances.

Political Context

Political contexts, institutions, and authorities are likely to constrain police behavior, including their use of protest policing tactics, in various ways (See della Porta 1996 for an additional review.) Broadly, political cultures and ideologies shape citizens expectations of police and the government officials they elect. Police in liberal democracies are expected to do different work than police in authoritarian regimes. Multiple studies have found that

left-leaning governments tend to be more tolerant, gentle, and consensual in their dealings with movements (Fillieule 1997: 335-40; della Porta 1995; Geary 1985: chapter 7; Winter 1998). But this is not always the case. Left-leaning governments will sometimes deploy 'hard' protest policing tactics as well (Funk 1991), leading scholars to surmise that they are responsive to electoral pressures to demonstrate their 'law and order' *bona fides* (della Porta 1999). ***This dissertation follows Earl and Soule (2006) in operationalizing the political climate in which police operate as the '% vote for the Democratic candidate in the most recent presidential election' to test the proposition that a cities' political ideological climate affects protest policing.***

Sometimes government influence on protest policing is rather apparent. McPhail et al. (1998) carefully traced the manifold federal interventions that forced a shift in US protest policing styles from 'escalated force' to 'negotiated management' of protests. Other times, government influence may operate indirectly. Waddington (1998) suggests that "institutional power is refracted through the lens of how police define their task." And since police tend to avoid 'in the job trouble' with superiors beholden to elected officials whose preferences are often easily imputed, those elected officials need not make their preferences explicit.

The form of a government may impact protest policing as well. Social movements scholars have argued that governments with multiple points of power (like cities governed by a council as opposed to a strong mayor) may be less-decisive, and therefore less helpful to movements (Huber, Ragin and Stephens 1993; Skocpol 1992; Amenta and Young 1999). Police departments may be strategically weakened, too, when government forms prevent political elites from taking clear decisions. On the other hand, less-decisive governments may allow police a free-hand to act as they please toward protest movements. Comparisons between Europe and the U.S. seem to indicate that additional layers of government tend to insulate police from elites. Thus, Earl acknowledges (2011) that elite opinions are "clearly important" to police in Europe (as evidenced by della Porta 1995; Wisler and Kriesi 1998) even though she argues elsewhere (Earl and Soule 2006) that they affect police relatively little in the United States. As Earl and Soule put it, "the peculiar institutional characteristics of police departments … independently structure the dynamics of protest control." ***This dissertation tests the extent to which government form insulates police from elites with an independent variable measuring the 'Number of power centers' for each U.S. city that experienced an Occupy encampment.***

Another way to test the effect of political elites on protest policing may be to observe whether police respond differently to protest when the political environment is unstable. Two competing hypotheses are relevant here. Police in cities that are reshuffling politically (due to elections) may be especially solicitous of elite directions. On the other hand, their behavior may reflect their sense that they are relatively free of supervision. ***This dissertation tests whether and how political instability effects protest by measuring police activity in light of a city's temporal distance from its most recent and upcoming elections.***

Contradictory findings regarding the effect of political context on protest policing may simply reflect the fact that government officials wield more or less influence over police use of specific protest policing performances in different countries, under different circumstances, and at different times. In other words, the research may be uneven because the influence of government officials is uneven. This dissertation will not just investigate if and how much political context and governance matters, but when and under what conditions it matters for each protest policing tactic on offer.

Police Capacity

Whether police are acting on direct orders from government officials, ignoring those officials, or something in between, their capacity to act is limited by the human and material resources at their disposal. A cash-strapped department will not be able to pay the overtime necessary to send large squads of police to observe a series of protest events recurring several days in a row. Other protest policing performances depend on physical technologies (like less-lethal weapons, riot-gear, urban assault vehicles and the like) that not all departments can access.

Despite the importance of these variables relatively little research has focused on the effect of police capacity on protest policing. Ron (2000) has shown that organizational factors affect militaries' abilities to repress opponents, and Boudreau's case studies (2001; 2009) have confirmed this finding.

Earl and Soule (2006; and Earl, Soule and McCarthy 2003) have led comparative research on this question. They found that police capacity, measured as per capita police spending,

increased the likelihood that police would show up to a protest (Earl, Soule and McCarthy 2003) *and* (2006) that better-resourced departments were less likely to use violent and illegitimate tactics (their so-called "Dirty Harry" approach). They surmised that "adequate levels of well-trained manpower may increase the chances that police are able to react in legal ways and without engaging in a police riot" (2006). One might *also* suggest that depleted departments are more likely to perceive movements as threatening to their officers or budget, and therefore more likely to seek to deter movements by responding with more force. ***This dissertation follows Earl and Soule (2006) by operationalizing Police Capacity as the per-capita-budget of a police department. It goes further, also measuring a department's number of officers-per-capita to test how these variables affect protest policing.***

Police Culture

Independent of the resources at their disposal, police departments may pursue protest policing performances consistent with their own departmental cultures and missions. James Q. Wilson documented the prevailing police cultures of the United States in V*arieties of Police Behavior* (1978), showing that police departments, like other organizations engaged in specialized activity, develop and reinforce solidarity through shared beliefs about who they are and how they relate to their wider community (Worden 1989). (See also Lundman 1980; and Skolnick 1966.) These group- and self-conceptions in relation to the wider community, what della Porta calls "police knowledge" (1998), may prefigure police understandings of their relationships to protesters. According to theory, departments that foster a sense of themselves as the keepers of law and order are focused on interdicting behaviors outside of normal routines and are likely to see protest as a nuisance at best, a threat to order at worst.

By this same rationale, this dissertation advances another police culture hypothesis: that departments routinely faced with violent crime are likely to spend less time, energy, and worry on protest movements. Readers are invited to consider how police perceive protesters of the middle-class left differently if those police work in a city that sees hundreds of murders per year vs. only a handful. In a city where violent crime is rampant, this dissertation hypothesizes, police are more likely to see their work as a daily, grinding effort to fight seriously traumatic outcomes. From the standpoint of a department that has developed such a culture, such a self-understanding of its work, a protest is likely to be viewed as only a distraction or a nuisance, not as a major disturbance requiring limited time and resources. ***This dissertation uses a city's violent crime rate (from the Bureau of***

Justice Statistics) to test the hypothesis that police in cities addled by violent crime will de-prioritize protest movements and commit less energy to their management.

While it is not always easy to measure police culture quantitatively, scholars have sought to test whether police training impacts protest policing behavior. Earl and Soule (2006) hypothesized that the number of "police studies programs" in a county might correlate with more 'professional' protest policing. They found that it did not, but this dissertation measures an aspect of police culture that may be even more likely than 'professionalization courses' to impact the ways police see their protest management work: a *department's dedication to community policing.*

Departments committed to a philosophy of 'community policing' habitually engage with community members as allies and may be more likely to view protest activities as legitimate expressions of community concerns. Fortunately for this dissertation, the Bureau of Justice Statistics has recently released data from its 2010 'Law Enforcement Management And Administrative Statistics' (LEMAS) survey that include a host of measures for community policing. ***This dissertation uses an index of community policing to test the hypothesis that a departmental culture of community policing reduces department use of force and violence against protest campaigns.***

Scholars have argued that other cultural effects may emerge from the demographic composition of police forces. Della Porta and Fillieule (2008, pg. 225) suggest a shift in Italian police culture resulting from the inclusion of women on the force. Given research indicating that police often operate through stereotypes of protesters (McClintock, Normandeau, Philippe, Skolnick, & Szabo 1974), one might hypothesize that a larger minority contingent among a police force (especially in a U.S. context) might also impact protest policing culture. ***This dissertation uses police department ethnic composition data to test the hypothesis that departments with a more diverse workforce use less force and violence against protest campaigns.***

Movement Characteristics and Activities

Of course, a major factor affecting protest policing is the movement itself. It seems obvious that authorities, whether government officials, police, or both, would deploy more aggressive policing performances against movements that threaten their interests. Plenty of evidence shows this to be true – from qualitative and quantitative studies across every inhabited continent and throughout history. (See Davenport 2007; or Earl 2011 for the thorough reviews upon which this section draws.)

A large amount of research, therefore, has sought to understand what makes elites and police feel threatened. McAdam (1982) has suggested that elites feel threatened by social movements rejecting institutional channels of policy change. Similarly, Bromley and Shupe (1983) argue that radical or revolutionary goals are most threatening. Such tactics and goals would certainly undermine the legitimacy of sitting governments. Even countercultural groups, though, seem to raise the alarms of authorities (Wisler and Giugni 1999). And authorities may become overwhelmed by groups seeking many changes at once (McAdam 1982) or from very large and active groups, whatever their cause (Davenport 2000). Even non-violent protest activities – marches, demonstrations, etc. – may appear threatening to police if crowds are large enough (Tilly 1978). In fact, as Eisinger writes, the implicit threat of mass disorder is a key feature of protest's power (1973, pg. 14).

Research on "threat perception" (Mahooney-Norris 2000) also suggests that states are aware of their own weaknesses (for instance, with demographic groups or in geographic areas) and that they may feel more threatened by movements that are comparatively strong where they are weak. A host of authors have argued more recently, however, that threats to police are much more likely to affect protest policing than threats to elites, especially in the U.S. context (Earl, Soule, McCarthy 2003; Earl and Soule 2009; Soule and Davenport 2006). (These arguments are more fully considered below.)

Other researchers have argued that weaker movements may also trigger the sorts of police use of force that can make headlines (Gamson 1975; 1990). Here, authorities are cast as bullies seeking only to engage in fights they can win. Duvall and Stohl (1983) make this argument in a broader state context as well. Collins (2009) proposes a psychological mechanism, "forward panic," that might explain violence against the weak and outnumbered as well. However, none of these theories have been quantitatively verified in the context of U.S. protest policing. Earl et al. (2003) used two measures of weakness and found no results confirming the theory. The first measure sought to test Piven and Cloward's (1977) notion that socially marginalized groups with few political allies were

especially likely to be victimized by police – a hypothesis in line with Stockdill's (1996) findings. The second sought to test the notion (in line with de Biasi 1998; and della Porta 1998) that media support for a movement might protect it from police repression. More research is needed in this area.

A Summation of Current Theory: Threatened, Reactive Police

Without exception, authors across the social sciences agree that a given protest policing response is likely to be caused by some combination of the broad factors above. There is less agreement, however, about how and when those factors combine to produce a police response – the subject of this dissertation. If there is any part of this literature where some consensus has emerged, however, it is in reference to protest policing in the United States.

American sociological theory on protest policing and social movements has been led, lately, by Jennifer Earl and coauthors Sarah Soule, John McCarthy, and Christian Davenport. While others have continued with qualitative work cataloguing evolving police responses, Earl and coauthors have delivered a raft of quantitative studies arguing for a reinterpretation of the second half of the 20th century of protest policing.

Along with others in the field, they have questioned the extent to which the shift from 'escalated force' to 'negotiated management' was ever as complete as McPhail et al. (1998) initially concluded. But while others have argued that states and police forces have become gradually more repressive, Earl and her colleagues have argued that protests have become more threatening, earning the more violent policing they have experienced. Moreover, Earl and colleagues argue that, at least in the U.S. context where most agree police are insulated from political elites (relative to their European counterparts), protesters are largely responsible for the police responses they get.

As Soule and Davenport concluded:

> While most viewed such events as the Battle of Seattle in 1999 and anti-WTO protests in Washington D.C. as an end of the détente between police and protesters, our findings suggest a different interpretation. Given the greater responsiveness of police to threatening protest, it is clear that such incidents of aggressive policing do not necessarily represent a throwback to an earlier pattern. Indeed, if we are right, then the only thing that had changed by the late 1990s was the manner in which protesters engaged in dissident activity. Both of these events were extremely large, were characterized by a diversity of tactics, and featured property damage—three of the factors found to significantly increase the likelihood of an aggressive police response. Thus, it is not so much that the police abandoned their philosophy of protecting protesters in favor of aggressively responding to them. Rather, it is likely that the features of these events were so threatening to police that they responded in the proportional manner that they have always done. (Soule and Davenport 2009, 17)

According to the authors, factors especially likely to affect police use of force against protesters included protester use of confrontational tactics like Rallies, and Demonstrations, and Civil Disobedience. In 71 percent of the 15,000 events they studied, protesters used these "confrontational tactics," bringing arrest and police violence upon themselves.

Readers might be forgiven for perceiving an apparent police bias in these studies' conclusions. In fact, Earl is very clear that she is attempting to insert a police perspective into the literature. It is her contention (and that of her colleagues) that such a perspective best explains protest policing in the United States where police are insulated from elites. Police in the U.S., Earl argues, enjoy:

> "high levels of discretion for line-officers in the performance of routine duties, historically varied attachments (and distance) from political elites, [and] an insular police culture cultivated formally and informally by police agencies."[1]

[1] One might counter that protest policing is not a routine activity, that varied attachments should be measured not ignored, and that culture, too, should be measured not ignored.

To best understand how police respond to protest, Earl argues that we must understand what they value and what makes them feel threatened.

> The social construction of danger and cues of danger means that not all objective risks are attended to by officers and agencies and that some objective risks are exaggerated (Tauber 1967). This institutional structuring of danger affects how police understand and relate to key institutional imperatives such as maintaining public order, controlling communities, and controlling interactions with the public (Balch 1972; Rubinstein 1980; Skolnick 1966; Tauber 1967; Wilson 1968).

Decisions about all policing, even protest policing, Earl would argue, are ultimately processed through a police habitus (Bourdieu 1977) preoccupied with control. Here, Earl quotes Rubinstein (1980):

> [The police officer] must learn to control his fears and anxiety by looking for signs of danger in the places and people he approaches; he must learn to examine people for signs of resistance, flight and threat, to limit their chances of hurting him or creating situations he cannot control or can control only with the use of force, which is appropriate to the circumstances. . . . He must accept and welcome the fact that, as a policeman, he must be in control of the situation lest it be in control of him (Rubinstein 1980, 75).

If police have any on-the-ground discretion in handling protest (and they surely do), Earl's Blue-centered approach offers a useful perspective to scholars seeking to understand the policing of protest. But the dedication of Earl, Soule, and Davenport to the perspective of police may go too far if it confuses readers into believing that police are merely reacting to protester threats in all cases.

The theory of protest policing advanced by Earl and her co-authors is a bold attempt to explain why police respond to protest as they do – the key questions of this dissertation. However, many aspects of the theory rely on unsupported assumptions and inferences. This dissertation uses more comparable, sensitive, and complete data to test the applicability of Earl's theory to police responses to the Occupy movement.

Some Concerns

Across their works, Earl, Davenport, and Soule argue not only that police violence is *correlated* with protester violence, property damage, and use of "confrontational tactics," but that it is *caused* by these protester activities. Here, they appear to overrun their data. Nowhere in any of their studies do they show or explain any method by which they ascertain that "aggressive" protester activities preceded the arrival of police on the scene or police's subsequent use of force and arrests. Their data merely show a correlation. In a footnote attempt to defend against this concern, Earl and Soule (2006) write:

> Critical readers may suggest that the police, at times, provoke violent actions or property damage by protesters. This would not affect our findings because police are unlikely to see themselves as provoking those actions and are also unlikely to change their reaction to violence or property damage even if they believe that their actions encouraged protester violence and property damage. Instead, police agencies might discourage action that would escalate the intensity of protests, but then allow significant police action once protests became violent or damaged property.

Earl and Soule appear to be stating that even when police provoke protester violence, they still feel threatened by that violence, and therefore the greatest predictor of police use of force and violence is still the threat police feel from protester violence and property damage. If police provocations of protesters are entirely unintentional, there is some logic to this. But if police intentionally provoke protesters, or even just arrive on the scene with a plan to arrest them, these authors' most weighty conclusions – that police only respond proportionally to protest events (as they always have) – seem to be undermined.[2]

[2] Beyond just "Seeing Blue," Earl seems to look the other way at the possibility that the police approach she calls "Dirty Harry" could include police-initiated violence. According to her findings, the chances of police using that approach only increase once protesters are *already* engaged in property damage and violence, *especially against counter-demonstrators.* One might wonder if, in fact, a "Dirty Harry" sort of approach might not include police accommodation of counter-demonstrator scuffles.

Earl and Soule (2006) also make pains to explain away elite interest in law and order, in order to support their argument that police independently react to protesters.

> It is unclear why political elites would have a particular interest in controlling missile throwing. However, it is clear that missile throwing directly threatens officer safety and was, during the period we examine, widely argued to be an indicator that police control was either lost or nearly lost over a situation.

Readers might also question the operationalization of concepts like "Confrontational Tactics" and "Radical Goals" that so threatened police according to these authors. One wonders why the following categories were not broken down to smaller categories given that the authors were examining 15,000 cases of protest events:

> **Confrontational Tactics**: We consider the following forms of protest to be confrontational: Civil disobedience, physical attacks, riots, mob violence, strikes, slow-downs, sick-ins and other conflicts. **We consider the following activities at protest events to be confrontational:** general civil disobedience, sit-ins and derivatives of sit-ins (e.g. shop-ins, penny-ins, etc.) physical attacks, verbal attacks or threats, blockades by protesters, building take-overs, looting, damaging property, kidnapping, and meeting disruptions.

> **Radical Goals.** We coded the following goals articulated at protest events as radical: **For:** comparable worth, ERA, socialism, communism, fascism, welfare, freedom of speech, Affirmative Action, minority political power, minority culture or pride, Black separatism, minority extremism, gay rights, Native American rights, and farm worker rights. **Against:** discrimination in employment for any minority group, the Vietnam war, imperialism, capitalism, the U.S. government, ROTC and campus military recruitment, government surveillance of protesters, the current status of minorities in America

More Concerns

If readers of Earl, Soule, and Davenport accept that police use of force and violence is never pre-planned and only results from the threat they feel from protesters, they might also accept their argument that American police are so independent from elites that scientists need not measure it. In fact, across all of the studies performed by these authors, there is almost no accounting for the political context in which police are acting. Soule and Davenport (2009) include no measures of political context in their models. Earle, Soule, and McCarthy (2003), likewise, include no measures of political context in their models. And Earl and Soule (2006) only include a measure of political liberalism as an unreported control variable in their models.

Also, while Earl and Soule (2006) acknowledge that features of police departments are likely to impact protest policing, this team of scholars has yet to find many measures of police capacity or culture that bear on protest policing. Earl and Soule (2006) and Earl et al. (2003) found that police budget reduces the likelihood of forceful protest control, but that is all.

More Research is Needed

To summarize, the quantitative sociological literature on protest policing argues that the protesting of police is mostly determined by protesters themselves. If protesters were to ask how they could avoid being beaten by police, these authors could only respond by advising them to avoid protesting in favor of radical goals including "freedom of speech" or "minority culture or pride." They might also advise protesters to refrain from engaging in "civil disobedience" like "rallies" and "demonstrations." If protesters felt these restrictions were too onerous or that police were too forceful, the current sociological voices speaking on the subject would advise them to take it up with police directly, because elected officials do not handle such matters.

To many observers, these conclusions might seem especially discordant with our current times. Since they were published, crowds and police have struggled through the Occupy movement and through the anti-racist police brutality movement. Even if sociology's conclusions about protest policing in the United States were correct for the late 20th century, times have changed.

Perhaps for the period of 1965 through 1990, police were totally even-handed to an array of protest events, each independent of the other. Perhaps they reacted to each of these events as they came, without government oversight, without any plan other than to show up if things got "out of control." Perhaps U.S. police departments are so autonomous that social scientists need not consider more than the general liberality of the cities in which they are embedded when wondering how politics affects their decision making.

Even if that is all true, the sociological literature on protest policing still needs an update. The last large-scale quantitative data we have been able to access is over twenty years old. To their credit, Earl and colleagues also call for more research to bolster or challenge their findings:

> It seems reasonable to conclude with some concrete suggestions for future research. First, our analysis ends in 1990, but clearly an analysis of protest policing between 1991 and the present would be enlightening, especially because of some of the well-publicized events of aggressive policing discussed above. Does our speculation about the dynamics of police response to such events as the Battle of Seattle hold up to a systematic analysis such as we have conducted for the 1960-1990 era? Second, our research was conducted at the event level and predicted the occurrence of two different police strategies based on event characteristics and time. But, additional analysis should examine these general questions at other levels of analysis. For example, one might examine yearly or monthly or weekly counts of different policing strategies, introducing lags to examine how past policing strategies impact present ones (net of, and in combination with, protester threat). Third, research should examine the effects of various exogenous factors, such as the overall structure of political opportunities on police use of force and/or violence and arrests. As well, one might examine how such exogenous factors interact with protester-generated threat to affect policing. Perhaps it is the case that police respond in a less proportionate manner (or, in other words, they are more tolerant of protester threat) when the political system is more open to protester claims. (Soule and Davenport 2009).

This dissertation will move forward with all three suggestions. It analyzes data from the present decade; focuses on multiple units of analysis including events (but also actions, campaigns, and cities); and measures and analyzes the effects of political opportunity

structures on protest policing. Elsewhere, Earl, Soule, and McCarthy (2003) also suggest that:

> Researchers must move away from [police] presence/absence formulations of repression and toward more theoretically and methodologically sensitive conceptualizations of police action. Further, we must refine existing theories so that hypotheses can distinguish among the suspected effects of independent variables on different forms of police action.

This dissertation takes these suggestions and runs with them.

New Research for New Theory

The "Blue-centered" approach offered by Earl and colleagues is not without its merits. In particular, it focuses researchers attention on the perspectives of the people who are most proximal to protest policing, the police themselves. The notion that police, in addition to elites, perceive threats, and perceive those threats differently than elites, is important to all future research on this subject. The associated notion (of the "Blue-centered" approach) that police merely *react* to protesters, however, should not be treated as a conclusive theory. Based on the evidence reviewed above, it is merely a hypothesis, potentially valid at some times and in some places. An alternative hypothesis still thrives in the literature: that police (regardless of how closely they are tied to elites) act strategically. They make plans. They anticipate protesters actions. They selectively under- or over-enforce laws (something even Earl et al. 2003 acknowledges.)

In fact, the notion that police engage protest strategically is commonplace among researchers studying Europe. It is also a strong finding among those who have lately turned their focus to the policing of international summit meetings (See, especially, Fernandez 2008; Starr, Fernandez, and Scholl (2011); and della Porta and Tarrow 2012.) In general, small-N studies focusing greater attention on the detailed interactions of police and protesters are better able to suss out the strategizing of police. Surely, the 'strategic incapacitation' approach identified by Noakes and Gilham (2007) – including police

removal of protesters to irrelevant designated protest areas – is not some a-strategic, reaction of police to ongoing protester threats.

When it comes to identifying police strategy, units of analysis matter, too. If researchers only record information at the event level, like Earl and her colleagues (and then analyze that data statistically), they are unable to notice who initiated violent actions during events where police and protesters were both violent. The same data could describe a slow escalation precipitated by a protester screaming racial slurs, or a police blitzkrieg of an unsuspecting, peaceful demonstration. The latter, of course, would imply that police premeditated their attack, and, therefore, were acting strategically. But Earl's and her colleague's data cannot show it.

Strategic policing can also be recognized by including data at a higher temporal unit of analysis. Many social movements string events together into campaigns. By observing police and protester activities across events, over the duration of campaigns, researchers can learn how police responses vary over the course of those campaigns. If they vary for any reason other than protester activity, researchers have evidence that police are acting strategically, or at least in response to some input other than protesters' immediate threats.

This dissertation will observe police and protester actions at all of these levels of temporal analysis, especially the campaign and event levels. Unlike many quantitative studies in political science, too, it will observe activity occurring in cities, the unit of analysis at which (at least in the United States) elite control is most likely to impact local, on-the-ground activity. It also considers more variables describing police departments, uncovering variations in police capacity and culture that impact protest policing.

Three Hypotheses: Reactive, Strategic, and Threatened Police

This dissertation addresses two literatures simultaneously. To the broader social science literature on repression and protest policing, it clarifies how and when political context, police capacity, and police culture affect protest policing. To the American sociological literature that is currently focused on a police-centered view of protest policing, it tests three broad hypotheses: the reactive police hypothesis, the strategic police hypothesis, and the threatened police hypothesis. Each of these non-exclusive hypotheses seeks to explain protest policing from the view of those most proximal to it.

The Reactive Police Hypothesis

Police are rather independent from elites, and their protest policing behaviors, including violence, depend on protesters' actions.

If a strong version of this hypothesis is true:

> Police activities should not vary across variables describing the *political opportunity structures* of the cities in which they are embedded.
>
> When movements in different cities behave similarly, police from those different cities should behave similarly in their response.
>
> If departments react differently to similar protester action, those differences should be wholly explained by variables describing the police department, since departments, according to this hypothesis, are insulated from the effects of variables describing city political context.
>
> Furthermore, even if data reveal that police responses vary by departmental variables, we should see little evidence that the *timing* of police actions is determined by variables describing the department. That would suggest that the capacity or culture of a police department effected the timing of their performance, which would, in-turn, suggest that police were taking decisions to plan, put-off, or speed up responses, – all telltale indicators of strategizing.

Strategic Police Hypothesis

Police act strategically when faced with protest. They consider the elected officials who supervise them, take stock of their own department's resources, and act in line with their department's culture to limit the risk of disorder posed by protesters. They even take initiative at times.

If police act strategically:

> Data should show that their actions vary depending on the political context in which they are embedded.

When movements in different cities behave similarly, police departments may behave differently, either because they are responding to the desires of political elites; optimizing the application of their own department's resources; or enacting heuristics relevant to their own department's culture.

If police act strategically, data should also show that the timing of their actions is affected by city- and departmental-level variables. The cadence of their activity is not solely dictated by protesters' actions.

Threatened Police Hypotheses

Police responses to protest are significantly influenced by their perception that the protest threatens their sense of control.

If police action are significantly influenced by a sense of threat to control:

Police should respond with more force (arrests and violence) to larger protest campaigns.

Police departments that are relatively small compared to their populations should feel more threat from campaigns, exhibiting less patience and tolerance for them. If the 'Strategic Police' hypothesis is correct, they may even try to deter future activities with early shows of force.[3]

Departments that are accustomed to significant disorder and violence should feel less threatened and react less forcefully to protest campaigns.

Departments that embrace a philosophy of community policing should see protest campaigns as less threatening, and react less forcefully to those campaigns.

Readers may note that the 'Threatened Police' Hypothesis can exist happily alongside either the 'Strategic Police' or 'Reactive Police' Hypothesis. This is intentional. Regardless of police departments' willingness or capacity to plan, their perceptions of threats are likely

[3] Whether or not deterrence is effective (see Earl and Soule 2010), many scholars since Feierabend and Feierabend (1966) have argued that police seek to deter protest with shows of force.

to affect how they respond to protests. The individual elements of these three broad hypotheses, and more, are tested in Chapter 6 and discussed in Chapter 7.

Campaigns, Performances, and Strategies

The ideal research design investigating how city and departmental factors affect protest policing is impossible to implement. It would involve creating hundreds of identical social movements and planting them in hundreds of cities at the same time. Each movement would espouse the same goals, organize in the same fashion, draw from the same repertoires and proceed with the same sequence of activities (at least at first). As variations in police and protesters activities emerged scientists would know they resulted from factors related to the cities in which their treatment 'movements' were planted.

This ideal research scenario is far from the design employed by Earl and colleagues. Their research uses data on 15,000 social movement events across NY state. Those movements had different goals, used different tactics, occurred years or decades apart, and varied from one-off events to sustained campaigns. The unobserved variations in these 15,000 events result in very noisy data, making it difficult for researchers to extract clear signals about the effects of variables related to city political context (political opportunities) or police departmental cultural variations.

Furthermore, the treatment of all of these 15,000 events as one-off events (as opposed to campaigns) forecloses opportunities to study city and police strategies. If each event is treated, statistically, as a brand new phenomenon unto itself, city and police responses will also be treated as one-off reactions to brand new phenomena. Analyses of long-term policing trends (e.g. Davenport and Soule 2006) may show changes in police behavior over years or decades, but the data prevent any inference about whether police are learning and shifting approaches over the days and weeks of a protest campaign – signs of strategic activity.

Here and throughout, this dissertation will use definitions from the social movements literature led by Charles Tilly, Doug McAdam, Sid Tarrow and the many researchers who have rallied around (or wrestled with) their theories of political contention. Therefore:

> A *campaign* is a sustained, organized public effort making collective claims on targeted authorities. Unlike a onetime petition, declaration, or mass meeting, a *campaign* extends beyond any single event – although social movements often include petitions, declarations, and mass meetings. A campaign always links at least three parties: a group of self-designated claimants, some object(s) of claims, and a public of some kind. The claims may target governmental officials, but the "authorities" in question can also include owners of property, religious functionaries, and others whose actions (or failures to act) significantly affect the welfare of many people (Tilly & Tarrow 2007, 119).

The 184 Occupy campaigns investigated by this dissertation fit all of the criteria of the Tilly and Tarrow definition. Chapter 4 and 5 investigate the extent to which these campaigns meet the conditions of the ideal research design outlined immediately above – one able to tease out the ways city and police factors influence shifts in police activity over days and weeks.

Since the campaigns were not created in a lab then spread across the United States, this dissertation must measure and (statistically) account for their differences. It does so primarily by measuring the activities of the campaigns in each city, and their prevalence through time. So far, this dissertation has used the word "activities" to describe all the actions of protesters or police. But the word conflates activity across important units of analysis. So, again, following primarily Tilly (2008) (in this case), this dissertation, henceforth, will clarify protester activities across four levels, *actions, contentious performances, contentious gatherings*, and *campaigns.*

Action is easiest to define because it conforms perfectly to lay understandings. *Action* is the smallest unit of analysis describing protesters activities in this dissertation and is operationalized through the grammar of the English language. An action is a *verb* carried out by some *subject*. It may include an object that could be another person, and it may

include a clausal complement clarifying the action further. For instance, all of the following are actions, according to this dissertation.

> Protesters chant.
> Police arrest protesters.
> Police spray protesters with pepper spray.

These *actions,* in every case, occur within the context of some event (Tilly often uses the word 'episode') defined by researchers. An event, like an action, fits closely with lay understandings of the word. An anniversary party is an event. It starts at the designated time or when a quorum of revelers arrive, and ends when the revelers have left the venue. Researchers define events similarly in the context of social movements. They begin, usually at a designated time, but really once a quorum have arrived, and last until that quorum is no longer sustained. In nearly all cases, these boundaries are rather easy to pick out, because news reporters have already marked the beginning and ends of events in their reports. Following Tilly, this dissertation refers to events carried out by protesters as *contentious gatherings*.

While most *contentious gatherings* include many *actions*, those *actions* often tend to clump together. The *actions* of a candle light vigil – lighting candles, saying prayers, observing moments of silence, paying respects to the departed – are almost never found immediately alongside the actions of a protester picket. *Actions* tend to cohere into pre-planned, or at least well-known, *performances.*

Tilly uses the dramaturgical metaphor of *performances* to emphasize that contentious *actions* are often part of some larger, rather-scripted whole. Many jazz artists, for instance, *perform* Nina Simone's "Feelin' Good." Each puts his or her own spin on the song, even varying their deliveries across concerts. Yet the song is still recognized as 'Feelin' Good" because musicians hit many of the same notes in the same time as Nina.

In the social movements context, activists learn, enact, and improvise *contentious performances* like marches, rallies, demonstrations, vigils, and the like. Each enactment is unique, but the performances are relatively stable in so far as they include similar actors committing similar actions. As Tilly put it:

> Once we look closely at collective making of claims, we see that particular instances improvise on shared scripts. Presentation of a petition, taking of a hostage, or mounting of a demonstration constitutes a performance linking at least two actors, a claimant and an object [i.e. target] of claims (14).

This dissertation will operationalize *contentious performances* using an innovative automated clustering method that directs a computer to recognize clusters of *action* that cohere into stable performances. This method is fully explained in Chapter 3. In short, it uses the predictable structure of English grammar to extract *actions* from news texts about Occupy's *contentious gatherings.* Then, it uses a (very sophisticated sort of) clustering algorithm to identify coherent clusters of *action*, i.e. *performances*, occurring within and across those *contentious gatherings.*

Tilly was very interested in *contentious performances* because he saw these as the key site of social movement learning and adaptation, the focus of much of his life's work.

> Innovation occurs incessantly on the small scale, but effective claims depend on a recognizable relation to their setting, to relations between the parties, and to previous uses of the claim-making form [i.e. performance]" (14).

This dissertation's interest in studying *contentious performances* has more to do with understanding how that learning and adaptation happens on the scale of days and weeks, rather than decades and centuries. Though Tilly, sadly, was unable to get to a study of *contentious performances* within *campaigns* (as Kriesi (2009, 345) also laments), he clearly saw the importance in tracing the use of performances over time, even within the context of social movement campaigns:

> The particular path of contention affects what happens next because each shared effort to press claims lays down a settlement among parties to the transaction, a memory of the interaction, new information about the likely outcomes of different sorts of interactions, and a changed network of relations within and among participants (16).

These provisional "settlement[s] among parties," "new information about the likely outcomes of different sorts of interactions," and "changed relations among participants" are exactly the stuff of *campaign* strategy.

This dissertation will advance Tilly's final frontier. But, it is not primarily interested in discovering the learning and strategizing of Occupy's activists during their campaigns. It is interested in discovering whether and how police were learning and strategizing as they faced those campaigns. This task, fortunately, does not require a vast re-thinking of social movements theory.

Like Ginger Robbins dancing with Fred Astaire, police "pretty much do everything *protesters* do, just in reverse." Their activities, therefore, are operationalized by this dissertation in precisely the same way with slightly different terminology. Thus, police activity comprises *actions* that cohere into *control performances* that are enacted at *police-initiated events.* (For now, this dissertation only posits the existence of *control campaigns*, though future research may discuss this concept further.) Here, it is worth clarifying that *police-initiated events* (again, identified by researchers by hand) are events in which police initiate contact with protesters. They do ***not*** include police responses to ongoing *contentious gatherings.* Chapter 6 will describe and explain *control performances* more fully, but examples include police going to Occupy encampments to enforce various city ordinances through citations, dismantling those encampments peacefully, or raiding the camps with a large show of force and the use of "less lethal" weapons.

By keeping in mind that protester and police activities encompass multiple units of analysis – actions, performances, gatherings/events, and campaigns – this dissertation enables analyses sensitive to the fact that *actions* and *performances* are chosen in the context of protester and police understandings of how a *campaign* is playing out. Are we winning or losing? How will we fight the next battle? What can we do to surprise them? How can we overcome their strengths and minimize our weaknesses? How can we divide them? How can we win more bystanders to our side?

In a world where each event is a brand new phenomenon, the world of Earl and her co-authors, these questions are hardly relevant. But in the context of a campaign, these questions define success or failure. This dissertation makes it possible, for the first time in a quantitative study, to imagine how police ask and answer these questions; how much

their strategizing is influenced by the political structures constraining the elites who supervise them; and how much it is constrained by their own department's capacities and cultural understanding of their work.

Chapter 3: Data, Methods, and Research Design

In the last third of 2011, 184 U.S. cities and towns experienced an Occupy encampment, defined by the presence of at least two tents for at least two consecutive evenings in some publicly-accessible space. Many of these encampments included dozens or even hundreds of tents pitched for weeks at a time. The occupation campaigns posed real and perceived challenges to city governments and police that resulted in many police and protester interactions. Those interactions, ranging from accommodation to expulsion and arrest, were recorded by local newspaper, television, and radio media in each of the cities where the occupation campaigns occurred. This dissertation processes textual data from such media accounts to compare protester and police performances across the many U.S. Occupy campaigns and draw inferences about how city- and police-level variables affected the policing of those campaigns.

The innovative multi-step process preparing the data analyzed by this dissertation required thousands of hours of researcher effort. The goals of all this data processing have been to extract from textual accounts of Occupy events information about Occupy and police *actions*; to render those *actions* countable and comparable; to understand how those *actions* cohere into *performances;* and, finally, to understand how the prevalence of those *performances* vary over the course of Occupy campaigns and with respect to variables describing the political situations of cities, and the capacities and cultures of police departments.

This chapter will describe the dissertation's research design, situation its data collection and analysis approach in a tradition of news and event analysis, describe the data collection and refinement pipeline in detail, introduce the topic modeling method used to identify *performances*, and discuss the method of regression-based structural topic modeling upon which much of the dissertation's analyses are based. The chapter closes with a discussion addressing data concerns that have been mitigated through the dissertation's design and data processing.

Research Design

This dissertation asks and answers questions that have been pending in the social movements and protest policing literatures for decades. (See Chapter 2 for a review of these literatures.) The great weakness of both literatures derives from the fact that most movement campaigns are unique or appear in unique geopolitical or temporal contexts. It is difficult to compare the policing of different movements when they occur in different countries at different times, let alone when their aims, strategies, and tactics differ considerably. As a consequence, sociologists and political scientists have made little headway answering questions that compare across movements seeking to understand the circumstances under which protester or police activities differ.

This dissertation begins to answer some of these questions, addressing, in particular, the factors that give rise to different protest policing responses in the United States. These questions are only askable and answerable because the Occupy movement spawned so many similar campaigns across so many comparable U.S. cities. The campaigns were not identical; nor the cities. But they were comparable enough to begin to understand how city-level variables and police department factors affected the activities of campaigns and police departments, and their interactions.

In the analyses featured in Chapters 4, 5, and 6, questions of the following form are asked: (1) What performances (identified by SVO-amplified LDA) were occurring during Occupy campaigns' contentious gatherings or police-initiated events? (2) How did the prevalence of these performances vary through time, from the beginning to the end of each Occupy campaign? (3) How did the prevalence of these performances vary according to city and police variables likely to affect them? (4) How did the effects of city and police variables identified by (3) vary through time? And, (5) Do answers to these questions suggest that police merely react to protest movements, or do they act strategically to control them?

Answers begin to come in Chapter 4, when this dissertation begins to ascertain the degree to which Occupy campaigns really are comparable. If campaigns are unique and fundamentally incomparable they will not aid comparisons across cities, comparisons seeking to understand the factors determining police-initiated responses to the campaigns.

Fortunately, structural topic modeling of text units (describe in this chapter, below) describing Occupy campaigns' contentious gatherings, in Chapters 4 and 5, reveals that Occupy campaigns were quite comparable. Chapter 4 discovers a range of contentious performances common to the known social movements repertoire, plus a few more associated with occupation campaigns. Structural topic modeling, in Chapter 5, reveals that a range of city-level variables, argued across a broad a deep literature to impact social movement participation and activity, did modestly affect the degree to which Occupy campaigns engaged in various contentious performances. However, these variables – population size and liberality, government type, and political instability – did not greatly disturb each campaigns' fulfillment of a common sequential occupation campaign life course.

The discovery of this common life course in Chapter 4, and its confirmation in Chapter 5, is crucial. The conclusion that Occupy campaigns varied relatively little, allows researchers to trust that variations in the policing of those campaigns resulted primarily from factors shaping city governments (already listed) and police departments (like community policing culture, budget per capita, personnel per capita, and police experiences with chronic violent crime), not from the local Occupy campaigns themselves.

Comparisons to Similar Data and Research

The data of this dissertation are unique. But the process by which they were generated has drawn on practices that have been common to social movements scholarship and event analysis for some time. Many scholars have studied news events (Gurr 1968; Jenkins and Eckert 1986; Jenkins and Perrow 1977; Kriesi et al. 1995; Lieberson and Silverman 1965; McAdam 1982; Olzak 1992; Shorter and Tilly 1974; Spilerman 1970, 1976).

These, and other researchers, have usually taken one of two approaches to analyzing news texts. They have either identified events and their details by hand (Soule and Davenport 2009; McAdam & Su 2002; McCarthy & McPahil 2006; Olzak & Soule 2009; Franzosi, De Fazio, & Vicari 2012; Tilly 2008; King, Bentele, & Soule 2007; Davenport 1997; Della Porta & Tarrow 2012); or they have attempted to identify events and their details using rule-

based algorithms programmed into computers (Hammand and Weidmann 2014; Gao et al. 2013; Keertipati 2014; Kwak & An 2014; Leetaru and Schrodt 2013; Schrodt and Yonamine). The former approach produces valid data but is extremely time consuming. The latter approach is significantly hamstrung by the difficulty of identifying events, situations, or contentious gatherings in text.

Hand-coding approaches, often called content analysis, rely on the manual labor of researchers and their assistants to categorize by hand the words and sentences of documents that refer to researchers' phenomena of interest. When hand-coding large text corpora including thousands of documents or more, researchers face a tradeoff between coding for more detailed information about more variables or finishing projects within a reasonable time frame and budget. For instance, the Dynamics of Collective Action team spent an entire decade collecting information about 22 variables describing 24,000 social movement events in the U.S .[4] The team could have finished faster if they had sought information on fewer cases or fewer variables, but reducing either would have limited the utility of their data for scientific analysis.

Newer, automated approaches to the analysis of text, meanwhile, require far less manual work. 'Dictionary methods,' like those used by this dissertation to find and replace named entities and verbs with normalized actors and verbs, can count keywords of interest. "Bag of words" approaches treat documents as matrices of word counts (where each document is a row, each column is a unique term found in the entire corpus of documents, and each cell is a count of how often that unique term appears in the document), setting up useful analyses like TF-IDF (Sparck Jones, 1972), and topic modeling (Blei, Ng, and Jordan, 2003). Grammar parsing methods help researchers extract information from text as ordered by its syntactic structure. As in this dissertation, grammar parsers can be used to resolve coreferences and to identify actions encoded as SVO triplets.

These methods are valuable, and their thoughtful use should be encouraged. But, they cannot accurately identify concepts like events, situations, or contentious gatherings. Such text analysis work must be done by humans. Though ClausIE (Del Corro and Gemulla 2013), a module of CoreNLP this dissertation uses, accurately extracts subject-verb-object

[4] For information and documentation on the *Dynamics of Collective Action* data, go to: http://web.stanford.edu/group/collectiveaction/cgi-bin/drupal/

(SVO) triplets describing who is doing what to whom, there are no grammatical units that reliably correspond with the 'social situation' or 'event' in which those SVOs occur. The contextual information that would allow one to identify the 'contentious gathering' in which an action occurs is often encoded in natural language clauses spanning multiple non-contiguous sentences. That is why this dissertation relies on humans to first identify the non-contiguous text units corresponding with protesters' contentious gatherings or police-initiated control events.

Automated attempts to identify events in text have failed repeatedly. The Global Data on Events Language and Tone (GDELT) project is just the highest-profile example of projects using poor-performing automated event-identification algorithms (Leetaru and Schrodt 2013). GDELT's event identifying algorithm extracts just one "event" per news article, and it defines that "event" as the first subject-verb-object triplet to appear in the news article. But, subject-verb-object triplets (SVOs) do not accurately identify events. They identify actions. And most events encompass multiple actions. Scholars, including those on the GDELT team, have recognized this for some time. They just have not been able to devise a better automated method of identifying events since events are not reliably encoded in grammar and the words delimiting them often span multiple non-contiguous sentences.

GDELT project director, Philip Schrodt has also lamented that GDELT's automated event-identification algorithm misinterprets many events. For instance, WWII memorial ceremonies are identified as open conflicts between European countries currently allied with one another, and GDELT's event-identification algorithm reads major football/soccer "battles" as literal battles between countries (Leetaru and Schrodt 2013). Hanna's review of data from GDELT (2014) finds that its automated event-identification algorithm misses major events, too, such as a march in Washington, D.C. attended by 50,000 people.[5]

This dissertation uses a hybrid – human and machine – text analysis workflow that employs researchers to do the event-identification work that computers cannot, and algorithms to complete work computers perform rather effectively and very efficiently.

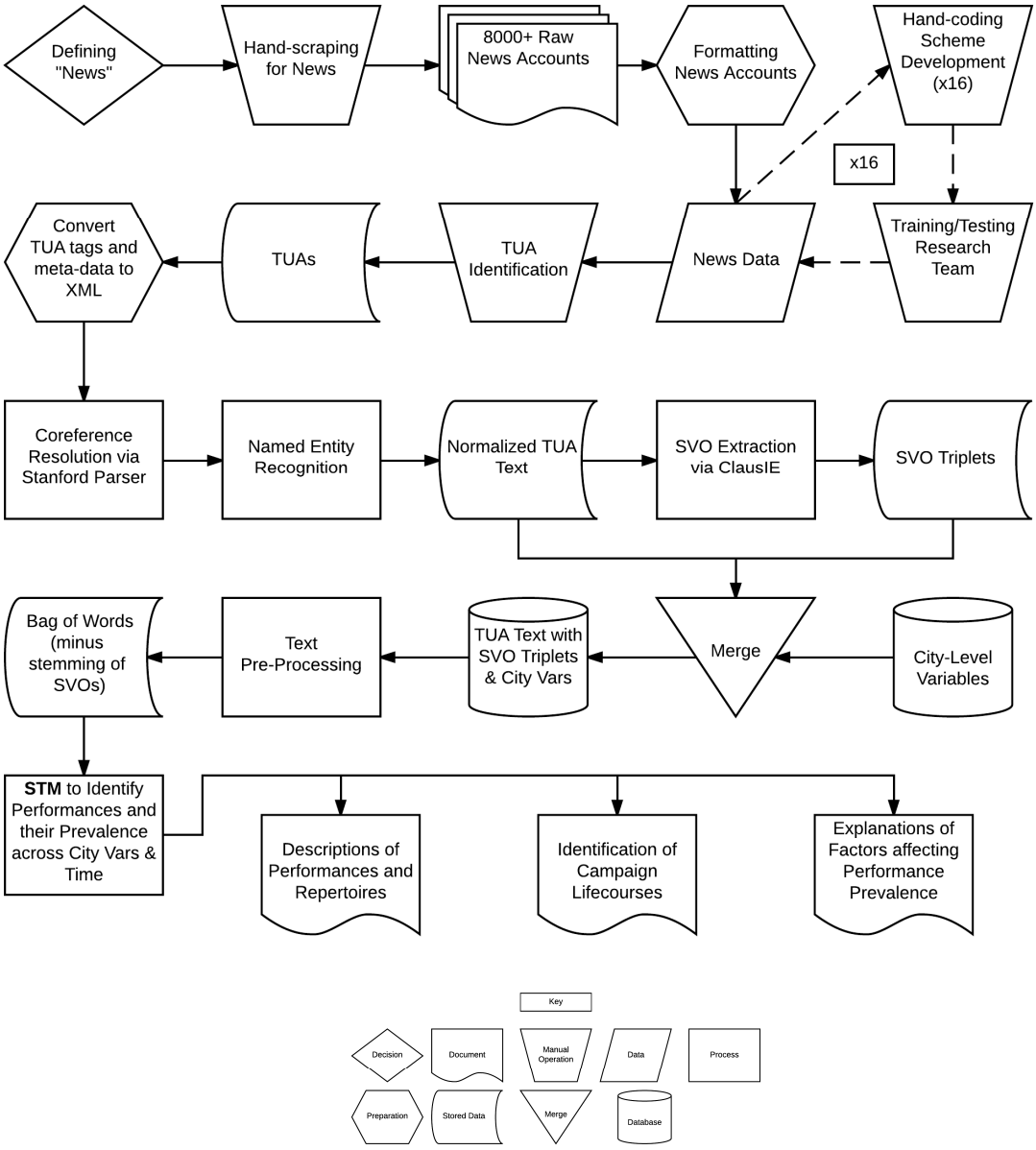
[5] See Hanna's computational social science weblog at: http://badhessian.org/2014/02/assessing-gdelt-with-handcoded-protest-data/

Data Pipeline

The data displayed in the many figures throughout this dissertation started as 8,342 news accounts describing protester, police, and city activities relevant to the Occupy movement of 2011. From these accounts, trained researchers extracted text units corresponding to particular events of the Occupy movement. These text units were run through algorithms of Stanford's CoreNLP (Natural Language Processing) to resolve coreferences and extract subject verb object information. Then, named persons in the text were transformed to 'protester,' 'police,' or 'city' and verbs were reduced to simpler forms using researcher-defined dictionaries. The normalized subject verb and object data in these text units were amplified using a procedure described below to ensure (in a way that compromises between Tilly's practice and Kriesi's (2009) wishes) that the *actions* of people are weighted relative to descriptions of event settings. Then, the underlying content of these text units was induced via Latent Dirichlet Allocation (topic modeling) to reveal coherent contentious and control performances in which protesters and police, respectively, engaged. And, finally, as shown in upcoming chapters, the prevalence of these performances was modeled as a function of city and police department variables like population and political liberalism, political stability, police capacity, community policing culture, and more.

This dissertation's data collection and processing pipeline is graphically displayed in Figure 3.1 below.

Figure 3.1: Data Pipeline



Note: TUA refers to text units of/for analysis, the hand-identified text units referring to contentious performances or police-initiated events. SVO refers to subject-verb-object triplets, actions describing who is doing what (to whom). ClausIE is a clause-based information extraction module of Stanford's CoreNLP, designed to extract SVO triplets. STM refers to the method of structural topic modeling and the R package used to perform that method.

Data Collection and Hand-Coding

With the help of a highly-trained team of undergraduate research assistants, this dissertation project collected every mainstream local, regional, and national news report on every Occupy campaign in the U.S. (The protocol used to find local news reports of the Occupy campaign is available in the Methodological Appendix A of this dissertation.) Next, each of these articles was stripped of formatting, dated (by its publication date), and placed in a Google Document folder corresponding with the news publication that published the account. These folders, in turn, were placed in a city folder containing all the news accounts for each city in which the particular Occupy campaign occurred. These meta-data (city of origin and publication date) enable analyses asking how the timing of events or the characteristics of the cities in which they were situated affected the activities occurring within them (described in the Research Design section above).

From the thousands of news accounts describing Occupy-related activities, trained researchers extracted 5,304 text units describing contentious gatherings initiated by protesters and 3,405 text units describing control events initiated by police. Each text unit describes only one gathering or event and excludes all other article text. The words and phrases composing the text unit are often not contiguous in the article. For example, the 3rd clause of sentence 5, sentence 7, sentences 9 and 10, and the last three words of sentence 15 could all compose one text unit that refers to a single protest march. Many articles included multiple text units, some about protest gatherings, others about police events, some about other units of interest that will be explored in other writings. (For an example of text unit identification by hand-coding, see Figure 3.2 below).

This dissertation focuses exclusively on contentious gatherings and police-initiated events. These events and their corresponding text units are defined not by the actors or actions within them, but by the actors initiating them. Therefore, a protester-initiated gathering may include actions by police that include warnings, the use of barricades, arrest, or more. On the other hand, police-initiated raids of encampments often include protester actions like fleeing or resistance. Researchers were trained to recognize text units describing

police- or protester- initiated events as intentional, pre-meditated constellations of activity that did not respond directly to the actions of the other party.[6] Police arrests of activists at an ongoing protest gathering, therefore, would be treated as activity within the protester-initiated contentious gathering. To clarify, regular encampment activities *not* associated with a particular rally, demonstration, march or other contentious gathering, were *not* treated as contentious gatherings. So, if police showed up to an Occupy encampment to warn protesters about a curfew, arrest key members, or expel the entire camp, these premeditated police activities were treated as police-initiated events. Such distinctions allowed the research team to preserving information about which parties were taking initiative to generate conditions that *might* precipitate protester and police interactions or confrontations.

The hand-coding of these text units, and thousands more describing other Occupy-related phenomena, required a year of effort over the Spring, Summer, and Fall of 2013. Text units describing police- and protester- initiated events (and other units of analysis) were hand-coded in two waves to ensure inter-coder reliability and data validity. Compared to hand-coding projects of similar scope, this duration is quite short. (The *Dynamics of Collective Action* project, for instance, required a decade of effort to collect data on a similar number of text units describing events.[7]) That is because hand-coding for this dissertation project did not require the extraction of variable and attribute data in detail.[8] Instead, this project uses automated text analysis algorithms in sequence to extract detailed data describing the activities occurring within text units describing contentious gatherings and police-initiated events. Figure 3.2 below displays an excerpt from a news article that has been hand-coded to identify three separate text units for analysis (TUAs).

---

[6] See Methodological Appendix A for further details about the hand-coding protocol.

[7] See The *Dynamics of Collective Action* project at: http://web.stanford.edu/group/collectiveaction/cgi-bin/drupal/

[8] Such detailed data will be extracted by crowd workers using an interface I am developing with the support of the Sloan Foundation, the Berkeley Institute for Data Science, The Digitial Humanities @ Berkeley, and the Hypothes.is Open Annotation Fund. See Appendix C for a full discussion of that work.

Figure 3.2: Hand-Coded Article with TUAs Identified

Amid chants of "Shame!" by demonstrators, ***State Police late Saturday made the largest group of arrests at the Occupy Albany site*** since THE PROTEST BEGAN OCT. 21.

"Warm up the bus," the protesters chanted as ***48 of their colleagues were arrested***, according to a news release issued by State Police early Sunday. In comparison, 24 people were arrested Nov. 12 the first time Occupy protesters ignored state instructions to leave the park before the 11 p.m. curfew.

This time, by 10:30 p.m. Saturday, about 75 protesters were in the park with no State Police presence in sight. However, more than a dozen troopers arrived at the scene shortly afterward. They issued their first arrest warning at about 10:55 p.m., five minutes before the curfew fell. A second warning that arrests were imminent was issued at about 11:05 p.m. After the second warning, some of the demonstrators dispersed throughout the park.

A circle of about three dozen protesters then formed, waiting for the State Police to arrest them. At the urging of those in the circle, several other protesters returned to join them and await arrest. … The troopers began arresting protesters at about 11:20 p.m.

[article truncated]

Note: Three TUAs have been identified in this article: a police-initiated event TUA in ***italicized bold font***; a protester-initiated event TUA in underlined font; and a second protester-initiated event TUA in UNDERLINED ALL-CAPS FONT. Each TUA corresponds with a single event.

Latent Dirichlet Allocation – Topic Modeling

For this dissertation, the hand-coded text units corresponding with events of the Occupy movement have been processed via an automated text analysis pipeline using multiple conventional natural language processing methods with two novel modifications. The resulting data are not only suitable for the questions and analyses of this dissertation, they have been processed very quickly without requiring the kind of painstaking hand-coding that sometimes stretchs on for years using traditional hand-coding. While learning to apply the automated methods used in this dissertation required many months of effort, anyone using the scripts written for this dissertation can now simply click 'run' and perform the same processing and analysis in about an hour (See Methodological Appendix A for these scripts).

All of the analyses in upcoming chapters rely on Latent Dirichlet Allocation (LDA). Here, I describe LDA, often called "topic modeling," before explaining the additional preparations that hand-coded event text units have undergone prior to being entered into topic modeling algorithms.

LDA is often used with text data as a way of uncovering the underlying subject matter in large bodies of text without having to read all of the text. Researchers feed an LDA algorithm documents and the parameter K, defined as the number of topics researchers believe are discussed by the text. LDA delivers two forms of output. First, it returns K different lists of words. Each list contains every single word found across all the inputted documents. But each list orders these words differently such that words listed first are most associated with one another and most exclusive of the words topping other lists.[9] These lists of words are called "topics", and when researchers read the first several words in the list, they gain an understanding of what the topic is about and generate their own label for the topic. This interpretive process will be familiar to those who have interpreted factor analysis output.

[9] Other term-weighting algorithms used in this dissertation, like FREX and Lift, privilege the terms most exclusive to a topic relatively more than the frequency of the terms in the topic.

LDA algorithms also generate a second form of output indicating the distribution of these "topics" in each document. So, each unit of text (document)– whether it is a news article, a paragraph, a sentence, a tweet, or a text unit identified by hand (as in this dissertation) – that has been fed into the algorithm by researchers can be summarized as some percentage of Topic 3, Topic 5, Topic 24, etc. When trying to make sense of thousands, let alone hundreds of thousands of documents, topic modeling helps researchers quickly understand the composition of their large body ("corpus") of documents as a mixture of several topics.

LDA is often called a "generative model" because it attempts to mathematically model the way in which a document is written or generated in the first place. The creators of LDA developed a generative model of writing by imagining an author setting about to write a document about some number of topics. The writer begins the writing process by asking herself, "What percentage of this document will be about topic 1, topic2, topic 3 … topic k?" Anyone who has outlined a dissertation (for instance) will recognize that one *does* often set about writing with some intention to focus different proportions of the document on the various topics it comprises. The hypothetical writer next inventories her vocabulary, a list of all the unique words she could use in the document (or across several documents) and then asks herself, "What is the probability that each of these words appears when I am writing about each of the topics in my list of K topics?" According to LDA's generative model, her writing process takes the following form: Beginning with the first word of the document and each word thereafter, she first rolls a K-sided die weighted so that the probability of it landing on any of its K, "topic" faces, corresponds with the distributions of each of those topics in the document, the answer to her first question. Given that the first word of the document is about Topic 3, say, she next rolls a second die unique to Topic 3. This die has as many faces as there are unique words in her vocabulary (defined as all the unique words in a corpus of documents). But this die is weighted according to the answer to her second question, so that terms most associated with Topic 3 are most likely to face up. Once these dice are cast, she commits the resulting word to her document and moves on to the next word of the document, performing the same procedure until the entire document is written.

Now, no one, especially the creators of LDA, believes that *New York Times* articles, or any other documents, are actually produced through such dice games. But the modeling of document generation in this way allows for the mathematical reverse engineering of the dice in our metaphor above – the probability distributions by which terms appear in topics and by which those topics appear in documents. Moreover, this model allows one to recover these probability distributions from "bags of words," document-term matrices

(where each document is a row, each column is a unique term found in the entire corpus of documents, and each cell is a count of how often that unique term appears in the document) that do not require anyone (or more to the point, a computer) to read words in the order they were written. Such a model makes it computationally possible to process large quantities of text, like the corpus on which this dissertation is based, in a reasonable amount of time.

The process whereby our metaphorical dice are reverse-engineered is very similar in intuition to clustering – a method with which many readers may be familiar. A simple K-means clustering algorithm arrays data along two dimensions (often eigenvectors of a larger matrix, or other dimensions specified by a researcher) and identifies K (an integer >1) clusters in the data, where K is also specified by the researcher. A K-means clustering algorithm proceeds by first assigning, at random, K points in the two-dimensional space as the centroids (centers) of each of the K clusters. The algorithm then assigns each datum to correspond with that centroid closest to it in the two-dimensional space. Next, the algorithm moves the previously randomly-assigned centroids to the center of the data assigned to it, maximizing the fit of the centroid to the data associated with it. Then, the algorithm iterates this process. Now that centroids have moved, it reassigns each datum to the centroid closest to it. Some data will keep the same centroid assignment while others will change. Then, as before, the position of each centroid is moved to the mean of all points assigned to the cluster associated with the centroid. This *expectation maximization* process, in which the algorithm designates an *expectation* of the final cluster to which each datum will be assigned and then *maximizes* the fit of the cluster's centroid to those data, continues until no more data assignments change, and centroids are no longer moved to maximize fit.

LDA, like K-means clustering, also uses an *expectation maximization* algorithm that iterates over these two steps. However, its expectation and maximization steps are a bit more complicated. In its *expectation* step, LDA generates a probability distribution specifying each term's likelihood of appearing in each topic – the second of our two metaphorical dice. In LDA's *maximization* step, it specifies each topic's likelihood of appearing in each document – the first of our two metaphorical dice. As with K-means clustering, these expectation and maximization steps iterate until there is no more change from one iteration to the next. At that point the algorithm is said to have 'converged' and a researcher can begin viewing the output described above: "topics" comprising lists of terms ordered by their probability of appearing in the topic, and a display of topic prevalence in each document of the corpus, or the corpus as a whole.

Though this approach to summarizing documents is still new to some scholars, it has been used widely and successfully since it was developed in 2003, and become especially popular with social scientists recently (Bonilla and Grimmer 2013; DiMaggio, Nag, and Blei 2013; King, Pan, and Roberts 2013; Laver 2000; Marshall 2013; Miller 2013; Mohr et al. 2013; Vavliakis, Symeonidis, and Mitkas 2013; Zirn and Stuckenschmidt 2014). As a new method, it is still somewhat untrusted despite its efficiency tackling large text corpora. Detractors often point out that the number of topics, K, must be arbitrarily chosen by researchers. However, this is not exactly correct. Topic modeling will seek to fit any number of topics, K, to a corpus of data. But if those data are not accurately described by K topics, a model will not converge. It will just iterate endlessly, never finding a fit to the data. The specification of K by researchers is probably best described as 'craft' as opposed to 'art' or 'science.' This dissertation, for instance used a K=40 topic model to uncover *contentious performances*. Wary of the pitfalls of data-mining, this dissertation used a single estimate, based on the author's deep reading of many text units, and the model converged quickly, so no other models were run.

For *control performances*, the dissertation first sought to topic model the corpus of police-initiated event TUAs using K=25. It was the author's estimate, based on a reading of many police TUAs, that police engaged in a smaller set of performances than protesters. However, the initial model did not converge. A model with K set to 20 did not converge either. It was only at K=15, that a model converged. This, in itself, is a finding (if not a surprising one): police engage in a narrower range of performances compared to protesters. But, it also demonstrates that the choice of K is not entirely arbitrary. Topic modeling seeks to fit a model to real data. And, while researchers must make decisions setting the parameters by which those models fit the data, they cannot do so by fiat. They cannot coerce *any* model to fit *any* data.

In this dissertation, LDA is used to summarize thousands of text units describing protester-initiated contentious gatherings and police-initiated events. But to increase the likelihood that LDA produces topics describing similar activity by similar parties across cities, these text units are normalized using some old and new natural language processing approaches.

Preparing Event Text Units for Topic Modeling

Many newcomers to topic modeling have been surprised by the quality of the output it generates (Grimmer and Stewart 2013; DiMaggio, Nag, Blei 2013). However, LDA is not without weaknesses. The method can be very sensitive to rare or exotic portions of text. For instance, in her dissertation work, Laura Nelson (2014) found that one of the topics generated by topic modeling hundreds of documents from women's liberation publications focused on car maintenance. The topic was extremely rare in the corpus of documents, but was so unique that it could not be 'clustered' with other topics that made up a larger proportion of the corpus.

Taking a warning from the experience of Nelson and others, this dissertation normalizes text units about Occupy activities in a few ways. First, since many of the people involved with Occupy campaigns bore different names, but similar positions, this dissertation uses custom dictionaries of named entities (people's names and/or titles) that are reduced to titles (like Police Chief, Occupy spokesperson, City Manager, etc.) and further reduced to 'police,' 'protester,' or 'city.' Across Occupy campaigns, these were the three major actors of consequence. The process of identifying named entities, called "named entity recognition," and transforming them into one of the three major actor classes of the Occupy movement was greatly aided by another common natural language processing technique: coreference resolution.

Coreference resolution is a process whereby pronouns are replaced with the noun to which they refer. If one inputs the sentence "Tanya likes ice cream and *she* likes pistachios" into a coreference resolution algorithm, it will be transformed into "Tanya likes ice cream and *Tanya* likes pistachios." By using Stanford Parser's (AKA CoreNLP's) coreference resolution module (Lee Chang, Peirsman, Chambers, Surdeanu, & Jurafsky 2013) prior to implementing custom named entity recognition dictionaries, this dissertation ensures that nearly all (CoreNLP's accuracy is very high but not perfect (Lee et al. 2013).) of the actors relevant to the Occupy movement, regardless of a news reporter's over-use of pronouns, were correctly identified with one of the three parties driving Occupy campaigns' dynamic interactions with cities and police.

With these two steps complete, a topic modeling algorithm could likely identify many topics of interest across the hand-coded text units corresponding with Occupy gatherings and events. However, inspired by the work of Charles Tilly, this dissertation adds another step to this text processing pipeline. Tilly (2008) recognized that reporters' language, particularly their use of verbs, was rather more artful than necessary or appropriate for scientific analysis. Whereas scientists try to clearly define concepts and use those concepts consistently, writers tend to find as many ways as possible to describe similar phenomena as they seek to keep their writing fresh and colorful. Tilly reduced over 2000 unique verbs appearing in news reports describing British contentious gatherings down to 34, verbs, then down to 8 verbs: claim, attack, control, cheer, communicate, deliberate, enter, and other. The corpus of 8,342 documents examined by this dissertation included 1260 unique verbs describing the actions in protester-initiated contentious gatherings. Seeking to normalize these 1260 verbs to aid topic modeling, but worrying that excessive verb reduction might do violence to reality, this dissertation reduces the 1260 verbs down to 464.[10]

All of the above text analysis procedures are deployed to increase the likelihood that LDA will produce topics highlighting the similar actions of similar actors across Occupy campaigns. Inspired by the work of Robert Franzosi, this dissertation has developed one other novel technique to increase the likelihood that LDA recognizes actors and their activities: SVO amplification.

SVO-Amplified LDA

Roberto Franzosi, since at least 1998 (Franzosi 1998), has been developing a method called Quantitative Narrative Analysis (QNA) that seeks to focus greater researcher attention on the individual actions and actors that compose events of interest. The method entails the hand-coding of subjects, verbs, and objects (SVOs)– who does what to whom – in text, and then the identification of these actions with the events in which they occur. The data generated by Franzosi's approach are extraordinarily rich, numerous, and close to the original text. Counts and graphical displays of actor/action networks enable intuitive analyses that Franzosi has combined to great effect with more traditional close readings. For instance, Franzosi's QNA (2012) allowed him to correct a history of post-Confederacy

[10] For an Appendix of verb transformation, please contact the author.

lynchings (Beck and Tolnay, 1990) that had failed to acknowledge the elements of racialized romantic jealousy and police complicity in White mob terror. Though painstaking, QNA achieves its namesake, allowing for the quantification of actions and story through time, and in a way that recovers the agency often lost by less granular quantitative approaches to events.

QNA's demonstration of the utility of SVO triplets as an analytical unit describing action inspired the method of SVO-amplified LDA used throughout this dissertation. As with QNA, SVO-amplified LDA requires the identification of SVO actions with the events in which they are embedded. However, the sequence of the process linking SVOs to events is reversed using this dissertation's approach. Instead of identifying SVOs by hand through a close reading, then identifying the events in which they occurred through a second close reading (Franzosi's procedure), this dissertation's approach extracts SVOs from text units that are already hand-coded to be conterminous with contentious gatherings and events. Using this dissertation's procedure, too, SVOs are extracted automatically using a new clause-based SVO extraction module of Stanford's CoreNLP called ClausIE (Del Corro & Gemulla 2013). ClausIE uses grammatical rules to identify the verbs in sentences and the subjects and objects of those verbs (as well as their open clausal complements when applicable).

Inputting the example police-initiated event text unit (for analysis) (TUA) from Figure 1 above into ClausIE produces the output appearing in column one of Figure 2 below. Once the coreference resolution, named-entity recognition, and verb reduction procedures detailed above are applied to that output, data appear in the form shown in column two of Figure 2.

Table 3.1: SVO Extractions From One Text Unit

Output From CoreNLP and ClausIE	**Simplified by keyword-based replacements**
***S**ubject_**V**erb_(verb negation)_**O**bject_Open Clausal Complement*	
the State Police_make_NA_the largest group of arrests at the Occupy Albany site_NA	police_make_arrests
NA_arrest_NA_48 of their colleagues_NA	_arrest_protesters

This time , by 10:30 p.m. Saturday , about 75 protesters_be_NA_NA_NA	protesters_be_
more than a dozen troopers_arrive_NA_NA_NA	police_arrive_
the 11 p.m. curfew_fall_NA_NA_NA	curfew_fall
more than a dozen troopers_issue_NA_their first arrest warning_NA	police_issue_warning
NA_issue_NA_A second warning that arrests were imminent_NA	_issue_warning
some of the demonstrators_disperse_NA_NA_NA	protesters_disperse_
A circle of about three dozen protesters_form_NA_NA_waiting	protesters_form_waiting
A circle of about three dozen protesters_wait_NA_NA_arrest	protesters_wait_arrest
several other protesters_join_NA_the urging of those in the circle_NA	protesters_join_the urging of protesters
several other protesters_return_NA_NA_join	protesters_return_join
several other protesters_await_NA_arrest_NA	protesters_wait_arrest
more than a dozen troopers_arrest_NA_protesters who remain in neighboring_NA	police_arrest_protesters
more than a dozen troopers_begin_NA_NA_arresting	police_begin_arresting

Note: 'NA' is returned when ClausIE does not find an 'Object' or 'Open Clausal Complement.' These markers are removed, as shown in column 2 above, prior to analysis.

These SVO triplets of column 2 are appended to the text units already identified by hand prior to topic modeling. The SVOs, therefore, are *amplified* in the text unit since they already appear once in the text unit. But, the SVO triplets, note, are conjoined with underscores '_' so that the computer, which uses blank spaces to recognize individual terms, will treat each SVO triplet as a single term to be counted just as other terms like 'riot,' 'plaza,' or 'banners.' With SVOs amplified in the text unit, LDA produces topics that are somewhat more focused on people and their activities than other incomparable information like weather, setting, and crowd size. Though these latter elements also figure in LDA's production of topics, SVO amplification increases the likelihood that topics will cohere around the activities of people determining the fate of Occupy campaigns.

As demonstrated in Chapter 4, SVO-amplified LDA performs. Many of the topics identified through the procedure call out the sorts of contentious performances – marches, rallies, demonstrations, and the like – that researchers have been identifying by hand for years. While SVO triplets, compared to more common words like 'protester' or 'police,' are rarely among the most frequent terms defining a topic, commonly used term weighting algorithms like FREX, which balances term FRequency and term EXclusivity (in constructing a topic) in its ranking of terms, and 'Lift,' which prioritizes exclusivity in its ranking of terms, accentuate the SVO triplets amplified by the above procedure. When interpreting topics by frequent terms *and* those exclusive to the topic, SVO amplification helps researchers to interpret the activities around which a topic coheres. The high-quality performance of SVO-amplified LDA owes substantially to the method's adequacy to theory. *Contentious performances* are composed of individual *actions.* SVO-amplified LDA privilege *actions* in the form of SVO triplets by appending them to the text describing all of the details of a particular *event.* LDA then uses the terms and SVO-triplets of that *event* to model topics that in many cases cluster co-occurring *actions* into *performances.* In the words of social movement theory, SVO-amplified LDA of text units bearing terms corresponding with protester-initiated contentious gatherings or police-initiated events produces topics that correspond with coherent contentious performances and control performances.

Structural Topic Modeling

Topics identified by SVO-amplified LDA are findings in and of themselves. They identify coherent and exclusive forms of activity – contentious performances and control performances – that may confirm or update researchers' notions of how protest campaigns and protest policing unfold. But these findings also set up additional analyses. Recall that each text unit of this dissertation bears meta-data identifying the city hosting the gathering or event to which the text unit refers, and the date on which the text unit was published. Recall also, that LDA output includes, in addition to topic term lists, a measure of the prevalence of each topic in each document of a corpus. Because each text unit is linked to a city, topic prevalence in a particular text unit (or aggregated across text units) can be ascertained for a particular city or a class of cities. This modeling of topic prevalence by meta-data hypothesized to *structure* topic prevalence is known as 'structural topic modeling.'

All of the ultimate analyses driving this dissertation use the method of structural topic modeling. In Chapters 4 and 5, structural topic modeling enables analyses showing the

extent to which different city-level variables shape protesters' contentious gatherings and contentious performances. In Chapter 6, structural topic modeling enables analyses showing the extent to which city-level and police-department variables shape the policing of protest through control performances.

Because each text unit analyzed in this dissertation includes the date on which it was published, structural topic models can also be used to understand how the prevalence of these performances varied over the duration of Occupy campaigns. Note, however, that a publication date is not equivalent to the date on which the event described by the text unit occurred. This dissertation uses the following, original, automated procedure to identify the date on which events described in text units occurred: First, the computer is programmed to search for the name of any weekday in the text unit. If the name of a day is found in a text unit, say 'Tuesday," it is next assigned the date of that Tuesday which occurred immediately prior to the date on which the text unit was published. If no day name is found in the text unit, the performances of the text unit are assigned the date one day prior to the publication date. Since all of the news accounts used in this data set derive from daily newspapers, radio, and television reports, this procedure accurately records the date on which the events described in text units occurred. (Note, too, that hand-coders distinguished text referring to future or planned events and these events were excluded from the current dataset.)

Assigning dates to the performances described in text units purchases a great deal of analytical power. With both city and temporal variables included and interacted in structural topic models, the analyses of Chapter 5 and 6 are able to estimate not only the degree to which city variables and time affect the prevalence of contentious performances and control performances, but also how the effects of city variables on performance prevalence vary over the course of Occupy encampments.

City and Police Variables

The structural topic models of this dissertation estimate the prevalence of protester and police performances as functions of city and police variables. Hypotheses relating performances to city and police variables are discussed more thoroughly in the literature review of Chapter 2 and empirical Chapters 4, 5, and 6. Here, I briefly describe the data sources and collection procedures for these variables.

Data on city populations was collected from the U.S. Census Bureau website. Data describing city and town government types was gathered primarily from the Municipal Year Book of 2011, published by the International City/County Management Association. Election data, including percent Obama vote in the 2008 election, and information about prior and upcoming elections were harvested from multiple local and regional sources. The source for each datum is stored alongside it.

Data on police variables including 'number of sworn officers per capita,' 'departmental budget per capita,' 'percentage of non-white officers,' and the index of departments' commitment to community policing philosophy were drawn from the 2007 Law Enforcement Management and Administration Survey (LEMAS) of U.S. police departments. These were the latest data available at the time of publication. One additional police variable, violent crime rate, was drawn from a more recent, 2011, Bureau of Justice Statistics (BJS) survey of police departments. Where departments were non-responsive to the LEMAS survey, researchers sought information from local websites and through direct email and phone contact with local departments. All of the sources for each datum are listed in a database that will be viewable to the research community once the embargo on this dissertation is lifted.

Data Concerns

Readers will note three concerns with the data of this dissertation: (1) News reports may not reflect reality. (2) Even if news reports do reflect reality, some events are reported multiple times by multiple sources, so data may indicate prevalence of event reports, not prevalence of events. (3) The analysis of performances over the life course of each Occupy campaign assumes that each campaign is independent of all the others. I address these three concerns, here.

The strengths and weaknesses of print news data have been thoroughly discussed by a range of researchers (Barranco and Wisler 1999; Franzosi 1987; Myers and Caniglia 2004; Ortiz et al. 2005) (See, in particular, Earl, McCarthy, & Soule 2004). Critics of news data are concerned with biases of omission and description. Research indicates (e.g. McCarthy et al. 1999), however, that journalists across sources can be relied upon to report the plain 'hard news' of events without many errors or embellishments. If anything, reporters are prone to

biases of omission. Such omissions are apparent in this dissertation's data. Local Fox news affiliates, for instance, reported Occupy events at less than 20% the rate of local NBC, ABC, and CBS affiliates.

The optimal strategy for overcoming biases of omission or description entails collecting data from as many sources as possible. This dissertation employs that very strategy, collecting articles on police and protester interactions for each city from Lexis Nexis and from all local, regional, and national news sources. Using the thorough news account collection protocol contained in Methodological Appendix A, trained researchers attempted to collect every news account of the Occupy movement that reported on events and activities of police, protesters, or city officials. When in doubt about a news source, these trained researchers included the news account in this dissertation's corpus since the goal was to be as comprehensive as possible. (Future work, described in this dissertation's conclusion will also include other event reports from independent media sources, weblogs, twitter, facebook, and police reports.)

Supposing that the news accounts collected for this dissertation comprehensively and accurately report the contentious gatherings and events of Occupy campaigns across so many cities, readers might still question the extent to which succeeding chapters' analyses measure the prevalence of contentious performances *in reporting,* not, strictly speaking, the prevalence of *contentious performances*. If some performances in some cities were systematically over- or under-reported, data may reflect differing media practices alongside differing protester and police activities. Such concerns have been anticipated and mitigated in this dissertation.

First of all, a skeptical reader might worry that accounts of Occupy activities in New York City and Oakland were over-reported and are, therefore, over-represented in the data. Indeed, those who followed the movement in the news will remember that these two campaigns garnered the most national and regional attention (in addition to the local media attention each individual Occupy campaign received). Fortunately, most of the reporting on Occupy events outside the region in which they occurred was very easy to identify since the vast majority of such reports went out through syndicated news feeds like the Associated Press and Reuters. This dissertation's hand-coders were instructed to hand-code duplicate articles (and even duplicate article paragraphs) for exclusion from the

dataset. That step alone has ensured that findings are not swamped by nationally syndicated articles that described Occupy campaigns in New York, Oakland, and other large cities. Because this step was taken, despite the fact that New York City's OWS campaign kicked off the Occupy Movement, it does not lead the dataset in numbers of text units describing protesters' contentious gatherings. It ranks third behind Oakland and Los Angeles (see Table 3.2).

Table 3.2 Selected Cities, TUA Counts, and POS Variable Scores

City	TUA count	Obama Supporters Score (1-5)	Political Instability Score (1-3)	Centers of Power Score (1-3)
Atlanta, GA	176	2	1	2
Baltimore, MD	25	3	3	1
Boston, MA	67	4	1	2
Chicago, IL	160	5	1	1
Dallas, TX	158	4	2	2
Denver, CO	122	3	1	2
Houston, TX	37	5	2	1
Los Angeles, CA	258	5	2	1
Miami, FL	51	2	1	1
Minneapolis, MN	104	2	1	2
Nashville, TN	62	3	2	2
New Orleans, LA	38	2	1	1

New York, NY	256	5	1	1
Oakland, CA	463	2	1	1
Philadelphia, PA	207	5	2	1
Phoenix, AZ	41	4	3	3
Pittsburgh, PA	69	2	1	1
Portland, ME	19	1	3	2
Portland, OR	126	4	1	3
Raleigh, NC	68	2	2	3
Sacramento, CA	29	2	1	3
San Diego, CA	155	4	1	1
San Francisco, CA	195	4	3	1
Seattle, WA	217	3	1	1
Tampa, FL	62	2	1	1
Washington, D.C.	95	4	1	3

Skeptical readers might also worry that cities with multiple newspapers and local radio and television news outlets produced many more reports per contentious gathering or police-initiated event than cities with fewer news outlets. News media in large cities, like New York, for instance, might report 5 versions of a protest march, while a small town like Brattleboro, VT with only one newspaper of record might report only 1 version of its local protest march on the same day. However, news outlets in smaller cities and towns often have less news to report, and are more likely to re-report news throughout the week. It is very common to see newspapers in slow-news markets re-tell the story of Tuesday's march as background to the story about Thursday's demonstration. This practice is less common in news markets with shorter news cycles where editors are seeking to cut, not fill, columns.

Regardless, when comparing the prevalence of contentious or control performance prevalence across larger and smaller cities (with more or fewer media outlets, respectively) prevalence is calculated separately for smaller and larger cities. That is, the prevalence of rally performances in small cities is measured as a proportion of all activities in *small* cities, not as a proportion of performances in all cities. This dissertation, therefore, adopts the modest assumption that the over-reporting of events, in general, is similar among smaller cities and towns, similar among medium cities, and similar among larger cities.

A skeptical reader might accept this assumption but still counter that media in differing cities may be systematically over-/under-report different *types* of performances. Indeed, one could imagine a reality in which some media markets, compared to others, are more likely to report spectacular events while ignoring smaller marches and demonstrations. Perhaps in larger cities, only spectacular protester and police showdowns make the news, while in smaller cities, any contentious gathering is a novelty interesting to reporters and their audiences. Or less innocently, media outlets in politically liberal environments might be more likely to cater to their audiences with news of Occupy (Franzosi 1987; Kriesi et al. 1995, p. 256), while publishers in conservative markets might help their audiences ignore their local Occupy campaign (Gitlin 1980; Herman and Chomsky 1988; Molotch 1979; Parenti 1986). The former bias is likely to be controlled for by a covariate specifying city size.[11] The latter, though, requires a covariate specifying the political liberalism/conservatism of a city. As sociologists studying the effects of urban environment on media reporting might note (Oliver and Meyer 1999), these two variables produce numerous downstream consequences shaping many aspects of cities including their media markets. Social movements scholars would point out, as well, that movements' mobilizations, numbers, and trajectories are all significantly determined by the size of their pool of potential supporters. Throughout this dissertation, therefore, analyses include a variable measuring the size of each cities' liberal community, calculated by multiplying the city's/town's population by the percentage of voters who cast ballots for Obama in the 2008 election.

But such statistical controls do not fully address the pointed critique that models estimating, for instance, the effect of political instability on the prevalence of contentious

[11] Future research will marshal more data from more varied sources including independent media to assess this.

performances Targeting City Hall (Figure 5.19 Panel 1) may simply reflect media's sense that these performances are more relevant nearer to elections. Moreover, this critique might include a concern that over-reports *amplify* media's biased interest in political conflicts in the run-up to elections. To address these concerns, spot-checks have been performed to determine if media systematically over-report particular activities and if they do so based upon POS variables like 'Political Instability.'

These manual spot-checks entailed the review of text unit content (whenever more than 5 text units were recorded for a single day of a local Occupy campaign) to ascertain if text units (a) were over-reporting the same performances on the same days, (b) if over-reports were systematically more common for some performances than for others, and (c) if those over-reports varied systematically along with POS variables like 'Political Instability.'

As expected, there was some over-reporting of activities since different news sources report on the same events and the same newspapers sometimes recall prior events. However, this over-reporting was not as starkly duplicative as one might think if imagining hand-coders counting events and defining their type by hand. Topic modeling does not work that way. It assigns topic proportions to each TUA, but each TUA is not 'counted' as this or that singular 'event type.' As a simple illustration:

5 TUAs about the same event on the same day often describe different aspects of the same event or describe the event in different levels of detail. But, these 5 TUAs are not quintuple counting an event as we would imagine if people were counting events by hand. Instead, these TUAs often include non-overlapping information, and each TUA is assigned different topic proportions where topics indicate the performances described in the TUAs. For instance, 5 TUAs about one march might describe the march in general, associated arrests, associated traffic battles, the march in detail, and some associated speeches. In such a case, the march may be "over-counted" somewhat, perhaps by a factor of 1.6, but not by a factor of 5. Much of the text across these TUAs is 'counted' as Arrests, Traffic Battles, and Speeches.

As a concrete example, consider the 10 TUAs out of Oakland in Table 3.3 below. All but one of these TUAs refer to activities occurring on October 24th, 2011 and describe what traditional hand-coders would describe as a single event. But each focuses more or less attention on the contentious performances that combine into that event. These TUAs do not over-count a single 'event' by a factor of 9. They describe many facets of a day's activities,

capturing the distribution of those activities more accurately than hand-coders would if they tried to treat the TUA as a single event to be reduced down to a description pre-defined by researchers.

Table 3.3 TUAs Describing Aspects of an Oakland Event

8:30 p.m. protesters toward city hall after scattering when police set off tear gas , protesters who have gathered again at frank ogawa plaza are protesting again , heading down broadway on their way back to oakland city hall . police are not following a crowd of protesters at 14th, but continue to stand sentry at frank h. ogawa plaza .	gather, march, police stand
protesters who have gathered again at frank h. ogawa plaza continue to regroup and are now returning to frank h. ogawa plaza , near where police deployed tear gas earlier . 10:10 p.m. protesters regroup, toward frank ogawa plaza	later, details, police stand
4 p.m. rally of occupy folks at 14th and madison occupy folks at 14th and madison 14th stayed calm in the early going . two buses filled with members of the santa clara police 's department in riot gear arrived on the scene just before 4 p.m. about 200 to 300 protesters were on hand chanting .	Earlier, chanting, police back up arrives
a small crowd of 40 gathered at the gated barriers erected to keep protesters out of the park in front of city hall shouting and spitting at the police line , protesters expressed their anger at what protesters witnessed in the early morning hours . outside city hall at about the same time, a rush of cops moved forward to prevent people into the park in front of city hall where protesters were established . one police with the number 87 on his helmet, pointed his rifle at the unarmed protesters screaming `` get back , get back ! '' a young man got on his knees before the police shouting , `` shoot a young man , shoot a young man ! '' breaking a tense moment , police pushed their way through the tangle of police and protesters, ordering the police to stand back in line and to secure the barrier with plastic ties.	numbers, city hall, police, tense moment
7 p.m. protesters pause at 20th and franklin for a rally in front of the california nurses association building. there are still at least 200 people in the street , blocking the intersection . at least three helicopters are	later still

circling overhead .	
protesters flooded the office of mayor jean quan and the alameda police 's office with demands to be cited and released.	target city hall
has occupied a tent city for two weeks . the loose-knit group occupied the plaza , repeating instructions over and over again two weeks ago	reference to previous camp establishment
estimates of protesters size throughout the evening fluctuated from 500 to 1,000 , as protesters and police began a march that wound from the downtown library to the city jail and then to city center.	numbers
protesters flooded the alameda police's office with demands protesters be cited and released.	demand release
500 protesters initially met at the main branch of the oakland library at 4 p.m. chanting that protesters would `` reclaim '' frank h. ogawa plaza , where protesters had been camped out for about two weeks	numbers, reclaim

While the proportion of TUAs carrying only duplicative information is very low, it is worth noting that these TUAs are often very short as well. Across the dataset, they usually carry very general summary information like “Protesters marched against corporate greed on Tuesday.” Since they are short, they do not significantly amplify the term counts upon which LDA is based.

To the extent that this amplification does occur, however, it may actually aid analysis. Future work will employ a strategy for linking counts of protesters to the performances in which they engage. But, on the frontier of automated text analysis, no one has yet devised a reliable way to do this. As a consequence, a march of dozen people *could* appear to be as significant as a march of 1200 people if they are reported upon at the same levels. If we adopt the reasonable assumption that larger events featuring *more* people engaging in *more,* and *more meaningful,* activity are more likely to be over-reported, this dissertation’s

text analytic approach to understanding police and protester interactions may actually benefit from some duplication of reports.

That over-reporting, a skeptical reader would argue, could be detrimental however if only some kinds of performances are over-reported, especially if they are systematically over-reported based upon the POS variables that are hypothesized to explain the prevalence of performances. A close spot-checking of the data has found that arrests *do* appear to be over-reported across the board. As other scholars have noted (Earl et al. 2004) arrest information is easy to come by for journalists (since it is a matter of public record) and it is often interesting to news readers. There is no evidence, however, that these over-reports vary systematically with this dissertation's independent variables.

But what of other performances? Do over-reports of those performances vary systematically with independent variables measuring political opportunity structures? For the most part, it is difficult to generate hypotheses imagining why political opportunity structures might influence over-reporting of specific performances. But, one can imagine that news outlets in cities nearing elections might play up those protest performances that directly target elites and (since our political process runs on money) their wealthy constituents. If readers direct their attention to Figure 5.19 Panel 2 in Chapter 5, they will notice a higher prevalence of reports highlighting contentious performances Targeting Banks in cities that were nearer to an election. In particular, there is a peak in Bank Targeting prevalence during the 18th-21st days of encampment. A skeptical reader will wonder whether this higher prevalence is driven by multiple events, or media amplification, and therefore, over-counting of events.

A close inspection of the data reveals that the higher prevalence, however, seems to be driven by increased Bank Targeting activity in cities nearing elections, particularly in Bangor, ME and San Francisco, CA, not mere over-reporting. If one, inaccurately, conceives of performance prevalence as 'event counting by hand-coders,' they might point out that activities in San Francisco were described by 6 TUAs when they could have been described by 5 TUAs. (One TUA offered only a short, duplicative, summary of a Bank Targeting event previously reported upon.) But given that San Francisco's activities featured over 500 protesters targeting four different banks over the course of four days, that skeptic would be hard-pressed to argue that media bias – as opposed to events on the ground – are driving the data displayed in Figure 5.19 Panel 2.

A more demanding skeptic will require more than a 4-day check of one contentious performance. "What about City Hall Targeting over the course of campaigns?," they might ask. "Why should readers believe that protesters are savvy and target politicians vying for votes when, perhaps, media just want to feed the flames of electoral controversy by over-reporting events that target city hall." A reasonable concern. However, of 19 TUAs describing City Hall Targeting in cities where elections were near, only 2 of those included duplicate information. Separate reports describing City Hall Targeting in San Francisco described how protesters marched to City Hall, that the SF Board of Governors was holding a meeting, that protesters interrupted the meeting to make demands, that the interruption amounted to "heckling'' (a report that was duplicated in another news account), that the mayor defended his decisions, and that the protesters numbered around 100.

Again, since this dissertation's analyses do not count events per se, but instead measure prevalence of *performances* in text, the duplication of the "heckling" report – especially since it comes in a single sentence – can hardly explain an over-counting of events substantially biasing findings. These checks (and others finding that multiple reports often included few, short duplications) allay concerns about systematically biased over-reporting. The incidence of purely duplicate reporting in the corpus is low. Multiple TUAs describing one event often describe multiple facets of that event. To the extent that some TUAs do repeat information, this does not result in a gross over-counting as would occur using a hand-coding method, but a modest amplification of (usually very short) summary information. That amplification is probably helpful given that current methods do not permit the counting of larger and more important events as larger and more important. But, apart from arrest reports, the over-counting does not appear to be systematic. While it may introduce some noise into the data (and even help observers see more relevant performances), a close investigation of the most likely and worrisome form of systematically biased over-counting has discovered no evidence giving cause for concern. Readers can trust that reports of performance prevalence map onto the prevalence of activities as they occurred during the Occupy campaigns of 2011.

The foregoing controls, and spot-checks bolster confidence in the findings of second-order analyses that directly compare the prevalence of performances across cities in different media markets. Other findings of this dissertation, however, require little or no reliance on the assumptions that over-reports are randomly distributed in the dataset. When studying the timing and sequence of Occupy campaigns' performances in Chapter 4, for instance, questions center not on the relative prevalence of performances from one city to the next,

but on the their timing and sequence over the course of campaigns. The analyses in Chapter 4 reveal a relatively common life course of U.S. Occupation campaigns – a stable sequence of different performances that, with only modest variation, occurred across the different cities experiencing an encampment. The timing and sequence of this life course is not revealed by perfect counts of events, but by the peaks and valleys of their prevalence through time. It matters not whether reports of protesters' curfew disputes with police averaged exactly 4 or 9 in number during the 6th weekend of Occupy campaigns. It matters that their prevalence clearly peaked during that particular weekend compared to all other weekends. The same goes for other performances during the Occupy campaigns.

On the subject of timing, this dissertation makes one final, strong assumption that might concern readers. When counting time, the analyses of this dissertation treat Day 0 as the first day of each *local* Occupy encampment. This decision entails the assumption that events of each Occupy campaign unfold according to local timing and dynamics. In many ways, this assumption should not be at all controversial. Movement campaigns are planned and carried out by the people in the places where rallies, demonstrations and the like occur. Campaigns contend with local city officials and local police. They draw support from local populations. They march through local streets and schedule their events in synch with other local happenings. Occupy, in particular, adopted an ethos that devolved authority to the most granular local levels.

However, some might contend that Occupy was a national movement and that time should be counted from the beginning of the initial Occupy Wall Street movement encampment in New York City. Indeed, some events – a bank transfer day, a day of marches, and a partially successful general strike – were coordinated through national social media to occur on the same day regardless of when local Occupy campaigns kicked off their encampments.

In fact, neither of these characterizations is entirely right or wrong. Occupy was a national movement with many independent local campaigns. But while the national aspects of the broader movement shaped the grievances, aims, and strategies animating activities on the ground, this dissertation is most interested in uncovering the ways that local campaigns interacted with local governments and local police. Those local interactions were planned and driven by local actors marshaling local resources and local supporters to contend with local governments and local police. Local governments and police, for their part, were responding in most cases to the real or perceived needs of their local constituents, drawing on their local cultural know-how and local resources to respond to local citizens setting up

encampments in local parks and marching through local streets. While the final closures of many camps were coordinated in time through a well documented conference call of mayors and police chiefs (Gold 2015), this fact does not at all suggest that city leaders developed policy preferences, out of whole cloth, on that conference call. It is much more likely that their independent experiences drove them to independent conclusions that the Occupations needed to end, and that the conference call only settled the moment when that end would come.

Media are wont to aggregate all of the local Occupy campaign narratives into a single national narrative, but consider the perspectives of activists, police administrators, and city council members in cities across the U.S. They were in the thick of this. They spent 40 or 60 or 80 hours a week either building a tent city within a city from nothing but enthusiasm and donated materials, or ensuring that buses ran on time, that garbage was being collected, that streets were repaired, that gang conflicts were managed, that housing projects were renovated, that public bonds were paid, that traffic was improved, that schools were in session, that neighborhood associations were appeased, the list goes on… *and* that the local Occupy campaign was managed within the boundaries of local constituents' expectations regarding 'law and order' and freedom of speech and assembly. To reduce the interactions of all these people to their mere enactment of suggestions appearing in the op-ed pages of the New York Times, Wall Street Journal, and Twitter is to ignore the fact that local life and governance both exist and demand a great deal of local attention and local energy.

Readers need not accept this dissertation's assumption of independent local Occupy campaigns in the absence of alternative data, however. Appendix B displays plots of all this dissertation's analyses performed using the alternative assumption that local campaigns were, in fact, only franchises of the original Occupy Wall Street encampment in New York City's Zucotti Park. These plots model the timing and prevalence of performances treating Day 0 as the first day of the Zucotti encampment. Readers may compare the plots of performance prevalence over time in Appendix B to those in the body of the dissertation which show rather clear signals (steep peaks and valleys) denoting a common life course of Occupy campaigns driven by local factors according to local timing. Then, they may determine for themselves the extent to which the movement was a national, centrally coordinated affair. (Future work, rather than relying on an assumption of local independence or national command and control, will attempt to specify, through a network diffusion model, the extent to which both local and national dynamics defined Occupy campaigns' performances.)

Chapter 4: The Occupy Campaigns and their Life Courses

Readers who were in the U.S. during the fall of 2011 will remember that there were days and weeks when one could not turn on the television or radio, one could not open a newspaper, without learning about some new development of the Occupy movement. Thousands were marching here. Hundreds were arrested there. In this city, a supportive shopkeeper closed his business to join the protesters. In that rural town, a college student home during fall break set up a lonely tent and banner at the corner of her parents wheat fields.

Interviews with protesters and supporters captured the heady energy of the time – eternally springing hope in some moments, frustration bordering on despondency in others. The peaks and valleys of these emotional narratives, like the rise and fall of Occupy activities and their interactions with cities and police, are well understood by some people for some locations. Consistent participants in the encampment in Cincinnati, Ohio, for instance, will still recall the surprisingly large rally that launched their campaign, their frustrations when the city denied their request for permission to camp, their peaceful acceptance of weeks of costly citations from police as they defied city ordinances, their successful demonstration targeting Fifth Third Bank, police arrests in the days after that event, and the disbanding of their camp in the face of so many exorbitant fines and arrests. Members of that movement will recall, too, their fortitude in the in the ensuing weeks, their determined efforts to mount legal challenges against the city, and their inspired re-encampment rally led by the Reverend Jesse Jackson. They will likely also recall their feelings of failure and hopelessness when police swiftly, seemingly effortlessly, shut down their encampment the very next day. Such detailed narratives, across all 184 U.S. campaigns, however, are little known throughout the scholarly community. They are little known even among the thousands or millions worldwide who participated in Occupy-related activities.

Even the storytellers reporting the Occupy movement to the public, be they participants or journalists, typically only knew a detailed account of what happened in the one campaign

they observed up close, and maybe the activities in larger cities, like New York and Oakland, that received a great deal of news coverage. This chapter will not report a timeline for each of the 184 Occupy campaigns in the U.S. But by gathering and analyzing data on the activities of all Occupy campaigns in the U.S., it asks a question that no one has yet been able to ask when observing only one or two campaigns at a time: *Was there a common pattern, or a few common patterns, to the rise and fall of Occupy campaigns' activities? Is there something like a coherent occupation campaign 'life course?'*

The notion of a life course is common in the fields of biology, ecology, psychology, and history. The concept implies that there is a 'natural' or 'normal' sequence and timing to the events that compose an entity's existence from birth to death. Life course analysis entails describing some typical life course among a population and the mechanisms that shape it, then identifying divergences from that typical life course that require explanation. Most people would agree, for instance, that it is accurate and useful to model a human life course as one that includes birth, physical growth, primary socialization, puberty, secondary socialization, achievement of physical maturity, physical decline, and death. Equipped with this knowledge and the expectations it entails, one can then notice that something is 'off' when, for instance, a 24 year old has not begun puberty, then potentially identify and study whatever condition or mechanism has blocked the (statistically) normal unfolding of that individual's life course.

This chapter begins a similar life course analysis for Occupy campaigns. The identification of an 'average' Occupy campaign life course is important for two reasons. First, the key questions of this dissertation concern how police respond to protest campaigns. In order to compare police responses to multiple campaigns across multiple cities, this dissertation must first ascertain that Occupy campaigns' activities were comparable. Certainly, the campaigns were motivated by similar grievances and tactics. But, to the extent that Occupy campaigns' use of tactics varied, the analyses of police behavior in Chapters 6 and 7 require that such variation be captured and controlled for statistically. These are the tasks of Chapters 4 and 5.

Second, the identification of an 'average' Occupy campaign life course is interesting in its own right to social movements scholars. Life course analysis is not entirely new in the study of social movements. Herbert Blumer introduced a theory of social movement "lifecycles" in the mid-20th century (1951, 203) suggesting that each movement underwent stages of "social ferment," "popular excitement," "formalization," and "institutionalization."

Many scholars rejected this theory, citing a number of movements that deviated from Blumer's model (Jackson and Morgan 1978; Meyer and Rowan 1983; Lowi 1971; Minkoff 1995; Zurcher and Curtis 1973). McAdam, Tarrow, and Tilly (2003) concluded that scholars should abandon the goal of fitting movements to a lifecycle model and focus attention instead on identifying the mechanisms and processes that shape social movements and other forms of contention.

More recently, however, scholars have picked up Blumer's ambition. Echoing Blumer, but allowing for more than one final stage (i.e., institutionalization), Christiansen (2009), defines four movement stages: "emergence," "coalescence," "bureaucratization," and "decline." Shultziner (2014) reorganizes these stages into three, yet broader, phases – "origins," "action," and "outcomes" – and goes on to suggest that the variables explaining each differ considerably. Origins often owe to social psychological factors. Actions are shaped by "structural and strategic factors" including the "interplay between the movement and its opponents." And, outcomes are determined largely by the strategic skill of movement leaders and their opponents. Since each of these stages is explained by distinct factors, Shultziner argues that the field of contentious politics would move forward faster if it analyzed movements according to these stages instead of attempting explanations that are "too wide," attempting to model whole movements with an expansive set of variables.

This dissertation works within Shultziner's agenda, addressing questions about the 'action' stage of social movements, and occupation campaigns in particular. Chapter 5, following Shultziner's recommendation to focus on those variables most likely to affect the 'action' stage, explores structural factors shaping protesters' deployment of particular contentious performances.

The Approach and Hypotheses

This chapter's exploration of the ebb and flow of various Occupy performances relies on a set of text analysis techniques, described thoroughly in Chapter 3, that may be new to some social movements scholars. These methods, especially structural topic modeling, enable the testing of hypotheses about the existence of contentious performances and their prevalence and sequences (in this chapter), and their variation in light of important city-level variables (in Chapter 5).

The analysis begins by testing the modest hypothesis that coherent contentious performances exist at all. Tilly (2008) argued in favor of this hypothesis (for his cases drawn from 18th and 19th century Britain) before showing how those contentious performances changed over time. His approach began with a principal components analysis of a matrix of verbs describing challengers' activities organized by the events in which they occurred. By plotting these in a factor space, he was able to 'eyeball' clusters of verbs that co-occurred in events. These verbs, he ably argued, cohered as the actions defining a number of contentious performances from petitions to marches and property destruction. This chapter uses a much-updated method, structural topic modeling (LDA) applying a similar intuition.

First, instead of extracting verbs from text and re-writing them by hand as Tilly and his team did (a practice risking validity and reliability error), this dissertation uses a grammatical parsing algorithm (described in Chapter 3) to extract subject-verb-object triplets (*who does what to whom*) from sentences about protester-initiated gatherings (e.g., marches, demonstrations, encampment activities, etc.). Then, instead of using a principal components analysis to set-up a researcher-intuited clustering of these triplets, this dissertation uses Latent Dirichlet Allocation (more commonly known as topic modeling) to allocate the action triplets into a number of coherent contentious performance 'topics.' (For a comprehensive description of these methods, see Chapter 3.) This approach reduces the impact of researcher error or bias. If coherent 'topics' describing protester actions emerge, they do so because the actions cohere in ways that even a computer (ignorant of all our sociological theories) can recognize. Moreover, the method is efficient. Work that would likely take years to do by hand, is completed in about an hour using LDA.

If the first hypotheses are confirmed, if topic modeling does discover coherent contentious performances from text units describing contentious gatherings, a second class of hypotheses may be tested. Because the text units analyzed by this dissertation include data identifying the timing of the contentious gatherings to which they refer, it is possible to test the hypothesis that the particular performances enacted at these gatherings were temporally non-random, that is, that they varied over time according to some ideal-typical (or perhaps a few typical) life course(s).

These common life courses, of course, are not totally independent of city governments and their policing of protest. Therefore, a third set of hypotheses – suggesting that the

prevalence of various contentious performances are related to the political opportunity structures in which they are enacted – are tested in Chapter 5.

This chapter's goal is to discover patterns in protesters' use of contentious performances over the life courses of the 'action' stage of their campaigns. But even before reviewing findings identifying contentious performances, I pause to hypothesize the sorts of contentious performances LDA might discover. Though topic modeling, requiring so little input from researchers, is surely a very *inductive* approach to the identification of contentious performances, it may be useful for a certain kind of hypothesis testing, the *sine qua non* of the *deductive* scientific approach. Specifically, based on close readings of several hundred news accounts about the Occupy campaigns, I hypothesize that a topic modeling algorithm will recover evidence of the stable and well-rehearsed 'social movement' performances that developed throughout the 19th century in Britain – marches, demonstrations, and rallies.

> *Hypothesis 4.1a-c: Topic modeling will recover evidence of stable and well-rehearsed 'social movement' performances including (a) marches, (b) demonstrations, and (c) rallies.*

In addition to these traditional social movements performances, the algorithm, is also hypothesized to identify other performances that were common to the Occupy movement like large-scale encampment and occupation of public spaces. This performance had not been common in the United States since 1932 when 17,000 WWI veterans and their families occupied a portion of under-utilized land in Washington D.C. to demand prompt payment of unpaid war-time bonuses.

> *Hypothesis 4.2: Topic modeling will recover evidence of contentious performances unique to urban occupation campaigns including encampment in public spaces.*

Most observers agree that Occupy's signature contentious performance, occupation, drew inspiration from the occupation of Tahrir Square by Egyptian students, Liberals, and opponents of the Mubarak regime. Indeed, the July 13th issue of Adbusters Magazine calling for the September 17th occupation of New York City's financial district specifically cites the occupation of Tahrir Square as a model to be emulated. The diffusion of this performance

to the U.S. has been described and analyzed by many popular and scholarly authors (Goodman 2011; Keating 2011; Gould-Wartofsky 2015). But, other (apparently new) contentious performances grew out of the U.S. Occupy movement as well.

U.S. Occupiers developed new contentious performances like the 'bank-transfer day,' featuring protesters encouraging customers of large and irresponsible banks to transfer their accounts to local banks and credit unions in order to punish and reduce the power of the very banks that precipitated the financial crisis of 2007-2008. Along these, lines, it is also hypothesized that topic modeling will recover frequent references to protester attempts to directly block access to banks.

> *Hypothesis 4.3a-b: Topic modeling will recover evidence of contentious performances targeting banks, including bank transfer days and the blocking of entrances to banks.*

A mature literature on social movements suggests that social movements' contentious performances often become more disruptive over the course of social movement campaigns. (See della Porta and Diani 2009 for a review.) Based on this literature and a close readings of hundreds of news articles about the Occupy movement, I hypothesize that protesters will engage in performances that use the tactic of blocking sidewalks and streets more often as campaigns progress.

> *Hypothesis 4.4a-b: Topic modeling will recover evidence of contentious performances blocking sidewalks and streets, and (b) these performances will happen more later in campaigns once performances have escalate to use more disruptive tactics.*

Once these contentious performances are arrayed across a temporal axis, a picture of life courses will begin to emerge. I further hypothesize, therefore, that the prevalence of camp establishment activities will peak early in campaigns, followed by traditional social movements performances, then more disruptive performances.

> *Hypothesis 4.5a-b: Topic modeling of performances through time will recover (a) evidence of an occupation campaign life course including (b) a sequence of activity*

beginning with camp establishment, performances of the traditional social movements' repertoire, and later more disruptive performances.

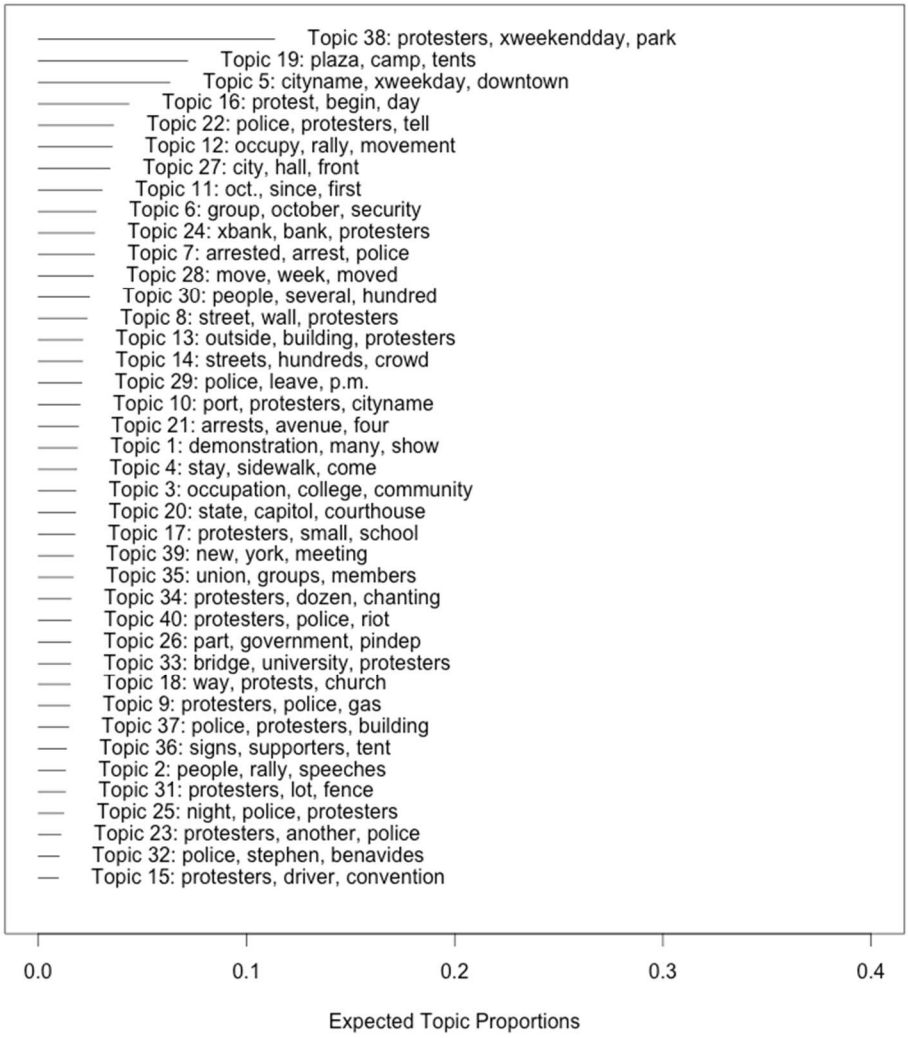
Results – Contentious Performances

Topic modeling algorithms generate two forms of output. First, they offer a list of ranked and weighted terms for each topic. Researchers interpret these term lists to determine what a topic is about. The algorithm also outputs a measure of the prevalence of each topic in each document. This output allows a researcher to understand what a given document is about without reading every last word of it. In this case, terms include information about *actions* that constitute topics of *contentious performance*, which, in turn, constitute text units describing *contentious gatherings*. Finally the prevalence of *contentious performances* across *gatherings* can be modeled through time (in this chapter) and in light of city-level variables (in Chapter 5).

The analysis of results begins with an interpretation of performance topics by the action terms they comprise. Then, analysis focuses on the prevalence of topic performances and types of contentious gathering through time – an analysis of how performances and gatherings do or do not constitute an Occupy movement life course.

Table 4.1, below, lists the top 3 most frequent terms for all 40 topics discovered using LDA topic modeling (described in Chapter 3). The line extending from the left to right indicates the prevalence of the topic in the entire corpus. Topic 38, about weekend protest activities, constitutes 11% of the corpus. Readers should take caution. This does not mean that 11% of activities related to the Occupy movement took place on weekends. It means that, according to the model, 11% of the reporting about Occupy protester-initiated events covered performances and gatherings that occurred on weekends. In general, data reported here are data on reports of events not events themselves. Some events may be reported multiple times while others are only reported once. This limitation of the data will be overcome in future work (see Chapter 7), but does not pose major complications to the analyses of this dissertation (See 'Data Concerns in Chapter 3 for more information). For now – as will be demonstrated below – reports of events, performances and gatherings still offer useful indicators of the relative rise and fall of Occupy activities through time.

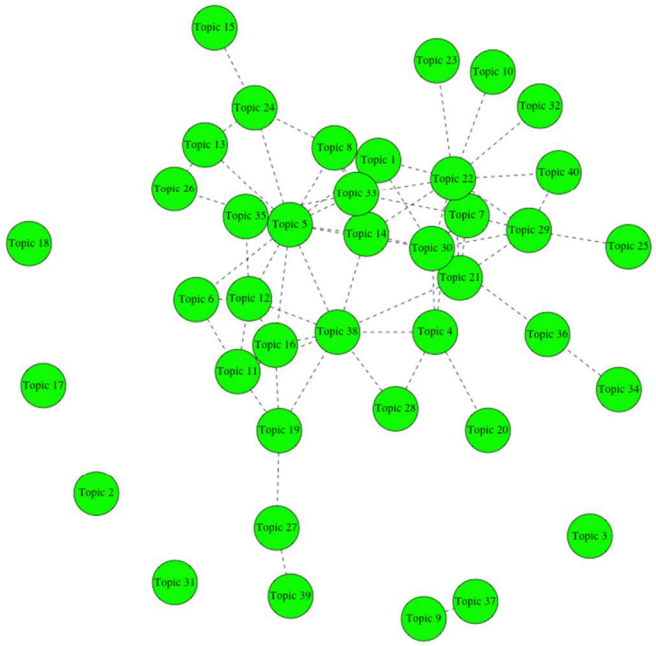
Table 4.1: Topic Terms and their Proportions in Corpus



Of these 40 topics, 13 are of particular interest. Their associated terms are listed in Table 4.2 along with a label I generated for each topic. (A complete listing of all topics' terms is available in Appendix B.) Of the 26 topics not included, many, like Topic 32 (about Dallas protest organizer Stephen Benavides) and Topic 15 referred to a very specific person or incident in a specific city. Such topics are useful in that they identify incidents that do not

generalize across Occupy campaigns, incidents that can be systematically excluded from analysis. Other topics, like Topic 2, seem, at first glance to be about general phenomena – "people, rally, speeches." But closer examination reveals that they referred, again, to some specific event. Topic correlations are visualized in Figure 4.1, below.

Figure 4.1: Graphical Display of Topic Correlations



Here, we see that Topic 2, along with Topics 31, 17, 18, and 3 are not closely related to other topics. Since they are both distant from other topics and occur rarely in the corpus, the analyses reported here treat them as rare events, incomparable to others and inexplicable by statistical approaches. They are ignored in the analyses here. Other topics are excluded because they are of no particular interest to researchers. For instance, Topic 11 is rather prevalent across documents and rather central in the network graph above. But Topic 11 captures sentences and clauses that inform news readers how long the local encampment has existed. Since researchers are interested in ongoing events, not several redundant reports of when a local Occupy campaign first established its camp, Topic 11 is

most useful in that it captures information researchers can easily exclude from analysis. Still other topics, like Topic 28 (about the relocation of camps) may merit future analysis, but are less relevant to the questions posed by this dissertation.

The analysis of the life courses of Occupy campaigns in the United States focuses on 13 topics discovered by the LDA model. Several of these topics, as hypothesized, cohere into well-known protest performances. Topic 1 represents activities in which "many" "show" "support" through the classic contentious performance called "demonstration."

Topic 5 represents "protesters" "marching," often in the "afternoon" and "evening" in some "downtown" location, and usually on "weekdays." Topic 12 represents "Occupy" "movement" "members" "rallying" in "numbers." In addition to the classic contentious performances – demonstrations, marches, and rallies – Occupy movements were notable for "gathering" in "parks" and "squares," more often on "weekends" (Topic 38), and establishing "Encampments" of "tents" in city "plazas" (Topic 19).

Table 4.2: Topic Term Lists for Topics of Interest

Topic 1 Top Words: ***Demonstrations***
Highest Probability: demonstration, many, show, showed, man, one, indep, smaller, handful, support
FREX: demonstration, showed, demonstration_, squid, latest, goldman, smaller, show, sachs, protesters_show_
Lift: _make_at, -take, alberta, authorization, bellagio, belmont, breitbart, canada, channelside, clipboard
Topic 4 Top Words: ***Sidewalk Contestation***
Highest Probability: stay, sidewalk, come, protesters_stay_, protesters_come_, eight, stood, remain, margaret, dozen
FREX: schucker, walden, dealings, speculative, wrought, margaret, protesters_stay_, attributed, stay, sidewalk

Lift: _allow_p.m., _allow_police_stay, _allow_some_do, _appear_allowing, _arrest_shucker_, _be_talk, _build_hours_, _damage_hours_, _decide_not/rb_happen_, _do_construction_

Topic 5 Top Words: ***Weekday Marching***

Highest Probability: cityname, xweekday, downtown, march, protested, afternoon, protesters_march_, protesting, around, evening

FREX: downtown, march, cityname, protested, xweekday, protesters_march_, afternoon, evening, protesting, sunttrust

Lift: _cite_protesters_disperse, _join_cityname, _march_the, _move_show, _show_solidarity_, 's_march_bring, afternoon_form_, ame, anti-drilling, attendees_hold_protesters_

Topic 7 Top Words: ***Arrests***

Highest Probability: arrested, arrest, police, protesters, -year-old, _arrest_protesters_, two, one, tell, trespassing

FREX: _arrest_protesters_, pregnant, arrested, -year-old, arrest, custody, comcast, linked, sit, trespassing

Lift: _arrest_anyone, _arrest_both_, _arrest_cityname, _arrest_eight_, _arrest_eleven, _arrest_five_, _arrest_four_, _arrest_members, _arrest_nine, _arrest_p

Topic 12 Top Words: ***Rallies***

Highest Probability: occupy, rally, movement, members, home, l.., local, rallied, number, national

FREX: occupy, l.., movement, rallied, home, foreclosed, protesters_rally_, rally, homes, okc

Lift: assn., bobby, bullock, carrefour, elk, hard-hit, hull, jan., legba, lifes

Topic 19 Top Words: ***Encampment Activities***

Highest Probability: plaza, camp, tents, set, encampment, two, center, weeks, camping, civic

FREX: protesters_camp_, _camp_protesters_, protesters_begin_camping, kitchen, tents, tents_, camp, weeks, camped, encamped

Lift: _camp_, _camp_occupy_, _camp_protesters_venting, _pitch_a, booths, city_spring_,

cityname_camp_, communal, crunchy, dewitt

Topic 22 Top Words: ***Traffic Battles***

Highest Probability: police, protesters, tell, polices, police_tell_, said, traffic, block, pepper, spray

FREX: pepper, spray, police_tell_, blocked, polices, police, spray_, bicycles, suspects, police_be_

Lift: _arrest_suspects_, _arrest_that, _close_broad, _dismiss_that, _dismiss_video_, _free_one, _injure_several_, _join_others_, _move_suspects_, _redirect_traffic_causing

Topic 24 Top Words: ***Bank Targeting***

Highest Probability: xbank, bank, protesters, branch, indep, banks, financial, california, close, district

FREX: pnc, institutions, coral, macdonald, accounts_, branch, bank, xbank, banks, financial

Lift: _arrest_bank, _arrest_nobody_, _arrest_sarah, _arrest_they_, _ask_protesters_wear, _bail_xbank, _chant_protesters_, _charge_six, _close_cityname, _close_his

Topic 27 Top Words: ***City Hall Targeting***

Highest Probability: city, hall, front, nov., city_tell_, lawn, xweekday, support, event, old

FREX: bagby, tech, hall, city, cityname-style, jazz, roommate, alive, duffie, jacket

Lift: alliances, angelenos, eaton, jeers, natalie, _allow_their, _ask_a, _ask_city_, _assign_the, _block_those

Topic 29 Top Words: ***Curfew Disputes***

Highest Probability: police, leave, p.m., park, arrested, protesters, arrest, refused, grant, tell

FREX: protesters_block_leave, grant, desks, leave, protesters_leave_, closing, refused, judge, congress, ordinance

Lift: balaklava, cta, innocent, _agree_the, _allow_dozens_continue, _allow_not/rb_the, _arrest_goodner, _arrest_hours, _arrest_its, _arrest_martinez_trying

Topic 35 Top Words: ***Labor Alliances***

Highest Probability: union, groups, members, jobs, protesters, indep, employees, labor, human, international

FREX: seiu, lori, kirby, dad, jobs, union, international, human, organizations, coalition

Lift: _arrest_jenn, _arrest_n't/rb_apd_, _ask_some_, _bind_n't/rb_the, _charge_jenn, _cite_several_, _claim_a, _confuse_protesters_, _dismiss_one, _drive_many_join

Topic 38 Top Words: ***Weekend Gatherings***

Highest Probability: protesters, xweekendday, park, square, gathered, protesters_be_, gather, xpark, second, take

FREX: xpark, xweekendday, park, gather, protesters_be_, square, protesters_gather_, gathered, protesters, second

Lift: _settle_, _tell_protesters_leave, angelia, argument_flare_, camping_march_, christians, city_occupy_protesters_, city-sanctioned, cityname_have_a, comfort

Topic 40 Top Words: ***Standoffs with Riot Gear***

Highest Probability: protesters, police, riot, gear, protester, move, one, deadline, cops, order

FREX: gear, rainey, deadline, riot, deputies, perimeter, standoff, tension, midnight, motorcycle

Lift: blaring, drigger, maalox, prez, uphold, _arrest_just, _arrest_protesters_blocking, _arrest_update_, _catch_protesters_, _charge_three

Note: Each Topic is represented by three lists. 'Highest Prob' lists the terms most frequently appearing in the topic. FREX lists terms that are frequent and exclusive to the topic (not used very frequently in other topics). 'Lift' lists terms that are most exclusive to the topic whether or not they are used frequently in the topic.

LDA not only captures all of these activities as topics, it also captures activities targeting particular entities like City Hall, banks, and commercial shipping. **Topic 27** identifies

"events" on the "front" "lawn" of **"City Hall"** and real-time city responses (i.e. "city_tell") to these events. **Topic 24**, meanwhile, highlights "protesters" attempts to "close" down **"bank"** "branches" and encourage their fellow citizens to "close" their accounts with those banks

LDA of protester-event TUAs even captured a topic describing the very important inclusion of Labor unions in the Occupy movement. **Topic 35** describes the participation of "members" of "union" and **"labor"** "groups" in "protesters" activities.

Though a full accounting of police responses to Occupy movement activities begins in Chapter 6 (the immediate analyses focus on events initiated by Occupy protesters), the activities described in LDA topics did not escape the attention or responses of local police departments. Consequently, LDA also modeled topics focusing on protester and police contests over sidewalk space, curfews, and street blocking, and topics about arrests and even standoffs with police in riot gear.

Topic 4 discusses whether police should "allow" "protesters" to "come" to, "stay" on, or "remain" on **"sidewalks." Topic 29** focuses on **curfew** disputes. As many observers of the Occupy movement know, "protesters" who "refused" to comply with "judge's" orders or city "ordinances" requiring them to "leave" "parks" in the "p.m." were often "arrested" by "police." **Topic 22** reveals that police showed even less tolerance for protesters attempts' to "block" or "redirect" "traffic." Such contentious performances often attracted multiple "police" warnings and even the use of "pepper" "spray."

As a result of these and other actions, police often resorted to "**arresting**" "protesters," the obvious focus of **Topic 7**. In other cases, police elected to show force with or without arresting protesters. **Topic 40** describes events in which "police" set up around the "perimeter" of "protesters" with "motorcycles" and "riot gear" and engaged in "tense" "standoffs," often in the anticipation of some "protester" "deadline" to follow a "police" "order."

The discovery of all of these topics suggests that LDA over TUAs can successfully identify topics/performances of interest to researchers. Next, this chapter explores the prevalence

of these topics over time to reveal if and how they constitute an Occupy movement life course.

Initial Hypotheses Confirmed

The discovery of contentious performances using topic modeling largely confirms the hypotheses above. Without effortful hand-coding, the text processing described in Chapter 3 was able to recover evidence of stable and well-rehearsed 'social movement' performances including (a) marches, (b) demonstrations, and (c) rallies, confirming Hypotheses 4.1a-c. Results also confirmed Hypothesis 4.2. Topic modeling identified contentious performances unique to urban occupation campaigns including encampment in public spaces. Topic modeling also confirmed Hypotheses 4.3a-b. It recovered evidence of contentious performances targeting banks. However, the method was not able to distinguish between bank transfer days and the blocking of entrances to banks. Finally, topic modeling also confirmed Hypothesis 4.4a, recovering evidence of contentious performances like blocking sidewalks and streets. None of these results are substantively earth-shattering. Readers should expect that these data appear in topic modeling. But, they are highlighted here to show the robustness of the topic modeling approach. Not only is it efficient, it is accurately identifies the sorts of contentious performances we would expect hand-coders to find through a much more painstaking process.

Results – Life Course Analysis

Life course analyses predict that activities of individuals, (here, Occupy campaigns) are not uniform through time, but ordered in some coherent sequence of stages. The analyses that follow, therefore, will test two null hypotheses and two more positive hypotheses. First, I expect analyses to reject the simplest null hypothesis predicting that Occupy activities are evenly distributed through time. A slightly less simple null hypothesis might predict that the activities of each campaign follow some sequence, but that these sequences are unique to each campaign and distributed such that no aggregate pattern of activity will be discernable. I expect the data, too, to reject this null hypothesis. More ambitiously, make the following hypotheses:

Hypothesis 4.5a-b: Topic modeling of performances through time will recover (a) evidence of an occupation campaign life course including (b) a sequence of activity beginning with camp establishment, performances of the traditional social movements' repertoire, and later more disruptive performances.

The life course analysis is aided by Figures 4.2 – 4.7, which visualize performance prevalence over time for a few topics at a time. For each of these visualizations, the time axis (x) begins 50 days prior to the establishment of the local campaigns' encampment and carries forward until 160 days after encampment started.[12] Readers will notice that each topic's prevalence in the corpus is visualized with 3 lines. The outer two lines indicate the bounds of the 95% confidence interval for each estimate of topic prevalence (the middle line) by day. Since fewer events occurred at the beginning and ends of each campaign, these confidence intervals are rather large for most topics/performances at the extremes of the time period under analysis. As a general rule for interpreting Figures throughout this dissertation, readers should ignore portions of the plots with very large confidence intervals (standard errors).

The equations used to estimate and plot predicted performance (topic) prevalence as a function of city and time variables are described in detail in the Methodological Appendix A and adhere to the assumptions of general linear regression models. The models behind Figures 4.2 – 4.7 estimate, predict, and display topic prevalence for the corpus of text units describing protester-initiated contentious gatherings. These models estimate $\beta_1, \beta_2, \beta_3, \ldots \beta_n$ using the following equation:

$$\gamma = \tau(x) + \beta_1(x) + \beta_2(x) + \beta_3(x) + \cdots \beta_n(x) + \varepsilon$$

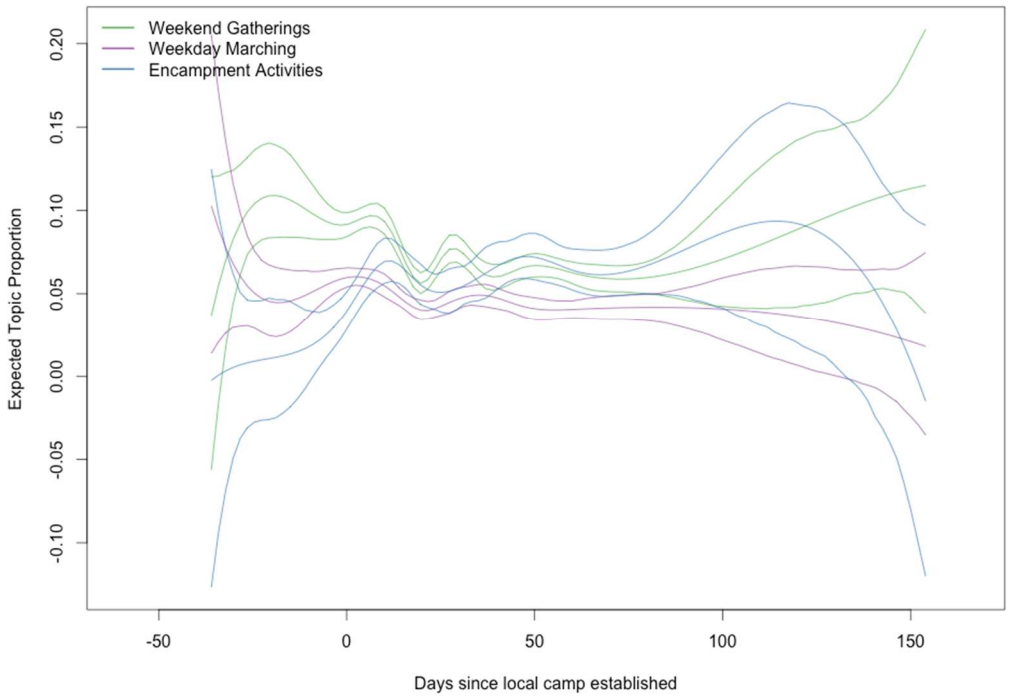
[12] The range of the temporal axis is defined by the data itself. All of the text units in the corpus describe contentious gatherings that occurred between 50 days prior and 160 days after the start of the local campaign.

Figures 4.2 – 4.7 then predict and display γ across (the duration of Occupy campaings) τ while holding $\beta_1, \beta_2, \beta_3$ (describing the number of Obama supporters, number of power centers, and stability of political alliances) at their means.

Life course analysis begins with a broad overview of Occupy campaigns' most common (or at least most commonly reported) activities. Figure 4.2, below, displays the prevalence of gatherings, marches, and encampment activities. As one might expect, reports of activities at Occupy encampments – including the institutionalization of regular General Assembly meetings, expansion of tents and dwellings, creation of Occupy libraries and kitchens, and more – peaked around the zero hour of encampments' starts, dipped momentarily and rose rather steadily over the duration of each camps existence.

Observing the prevalence of 'Weekend Gatherings' through time, it appears that most camps' beginnings – like in the case of Cincinnati, mentioned above – were preceded by large gatherings on the weekends prior to a camp's creation. These large events – rallies and demonstrations featuring speakers, local artists, musicians, and notables – continued throughout campaigns, but, on average, were overtaken in prevalence by regular camp activities around week nine of the encampment. The most stable of Occupiers' contentious performances is also the best established in the social movements repertoire: marching. Of all the activities protesters engaged in, marching was the one they performed with the most consistency from the beginning to the end of their campaigns. This third most prevalent performance (marching) was a mainstay of the movement.

Figure 4.2 – Weekend Gatherings, Encampment Activities, Weekday Marches



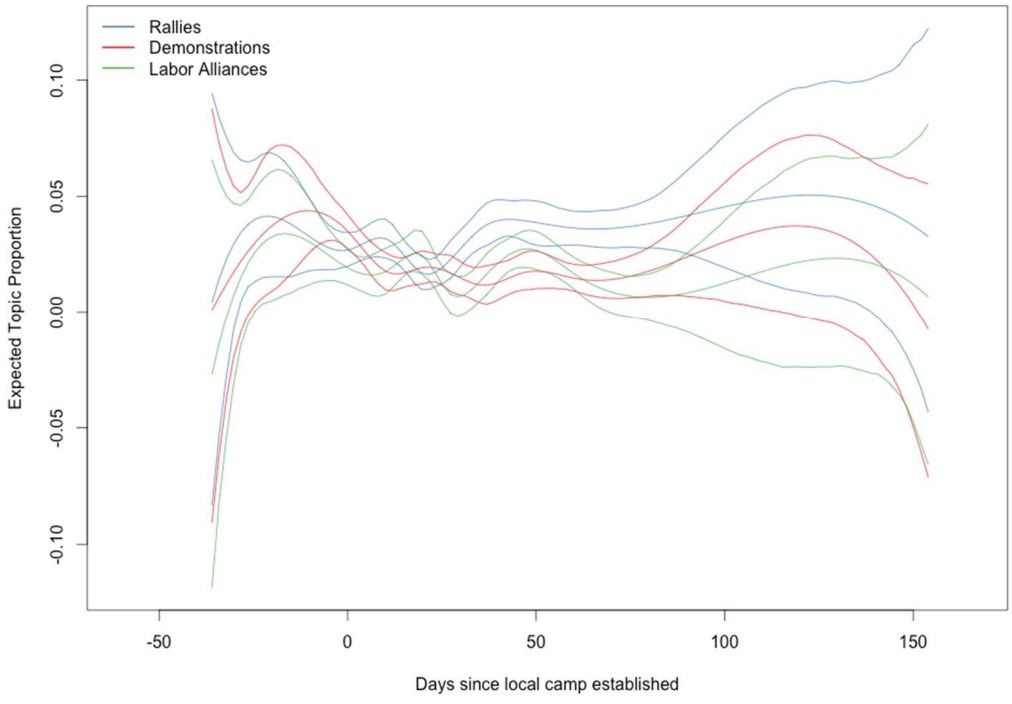
Note: Predicted prevalence of Weekend Gathering, Weekday Marching, and Encampment Activities performances by day of encampment across the full corpus of text units describing protester-initiated contentious gatherings. All POS variables (number of liberals, number of power centers, and political instability) are held at their means.

With this overview in mind, Figure 4.3 explores more specific contentious performances, Rallies and Demonstrations. Though these performances are rather similar in that they both feature protesters staying in one location (as opposed to marching), they differ in their intentions. Demonstrations are usually directed at some external target, often bystanders and passersby, sometimes particular government or media targets. The audience of protesters speaking and performing at rallies, on the other hand, is composed of other protesters. Rallies are designed to motivate and excite existing and new protest members, preparing them to engage in upcoming activities or further identify with movement goals. Given this motivation for rallies, it makes sense that they would increase in duration over the course of a campaign, as shown in Figure 4.3.

Demonstrations (or the reporting thereof) were most common near the beginnings and ends of campaigns. The use of demonstration at the beginning of a campaign to clarify claims and raise pressure on external targets does not beg for explanation. Given that many activities were being channeled into maintaining the life of Occupy encampments, it is not surprising, either, that demonstrations' prevalence attenuated after kicking off Occupy campaigns. The later surge in demonstrations is curious, however and its potential cause is explored more fully in Chapter 7.

If rallies, unlike demonstrations, are designed to invigorate the faithful, observers should expect that they increase in prevalence when new members are incorporated into the movement. This is, in fact, exactly what we see in Figure 4.3. Ignoring the noisy data prior to camp establishment (Day 0), readers will note a peak in the 'Labor Alliances' topic three weeks into Occupy campaigns. This is a moment, many will recall, when many labor unions were deciding whether and how to join forces with the Occupy movement. After some shuffling of feet, a number of unions joined their local Occupy campaigns in the succeeding weeks. By Day 50, many labor unions were shoulder to shoulder with Occupy protesters enacting a social movements performance they had rehearsed for decades: the Rally.

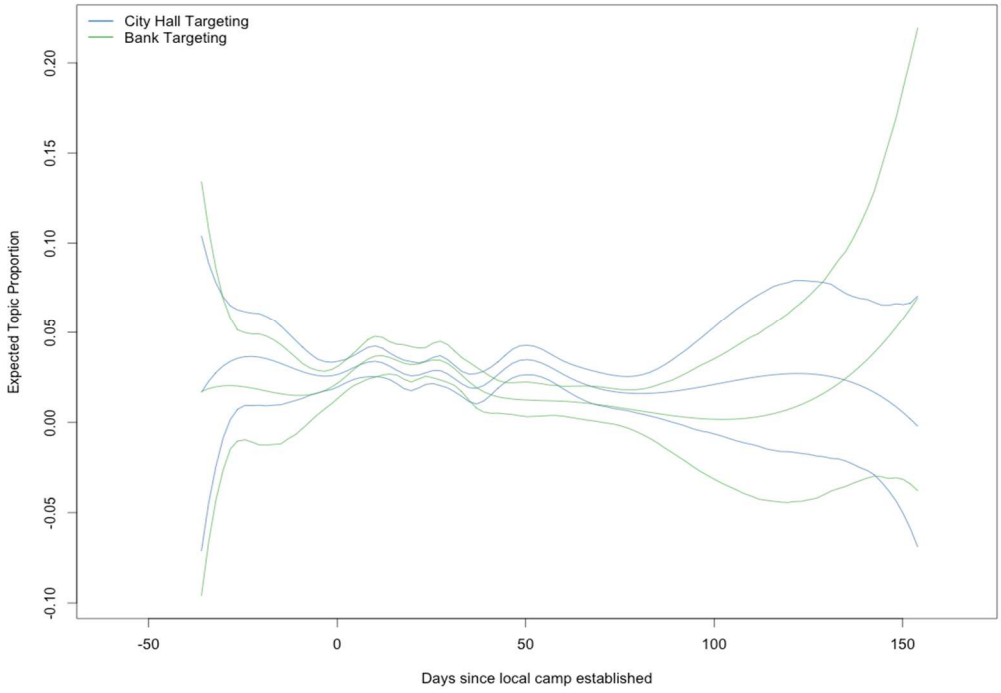
Figure 4.3 – Rallies, Demonstrations, and Labor Alliances



Note: Predicted prevalence of Rallies, Demonstrations, and Labor Alliances performances by day of encampment across the full corpus of text units describing protester-initiated contentious gatherings. All POS variables (number of liberals, number of power centers, and political instability) are held at their means.

Just as demonstrations are distinguished from rallies by the audiences they target, other contentious performances may also be identified by their targets. LDA discovered two topics of contentious performance by their association with their primary targets: City Halls and Banks. As shown in Figure 4.4 below, Occupy campaigns (in the aggregate) kept steady pressure on City Halls across the United States. Actions targeting banks were more prevalent near the beginnings of campaigns. In some cases, these actions involved protesters sitting in the lobbies of bank branches, or setting up picket lines immediately outside their doors. In others, protesters encouraged bank customers to transfer their accounts to local credit unions. In any case, these activities (or at least reporting on them) attenuated over time.

Figure 4.4 – City Hall Targeting and Bank Targeting



Note: Predicted prevalence of City Hall Targeting and Bank Targeting performances by day of encampment across the full corpus of text units describing protester-initiated contentious

gatherings. All POS variables (number of liberals, number of power centers, and political instability) are held at their means.

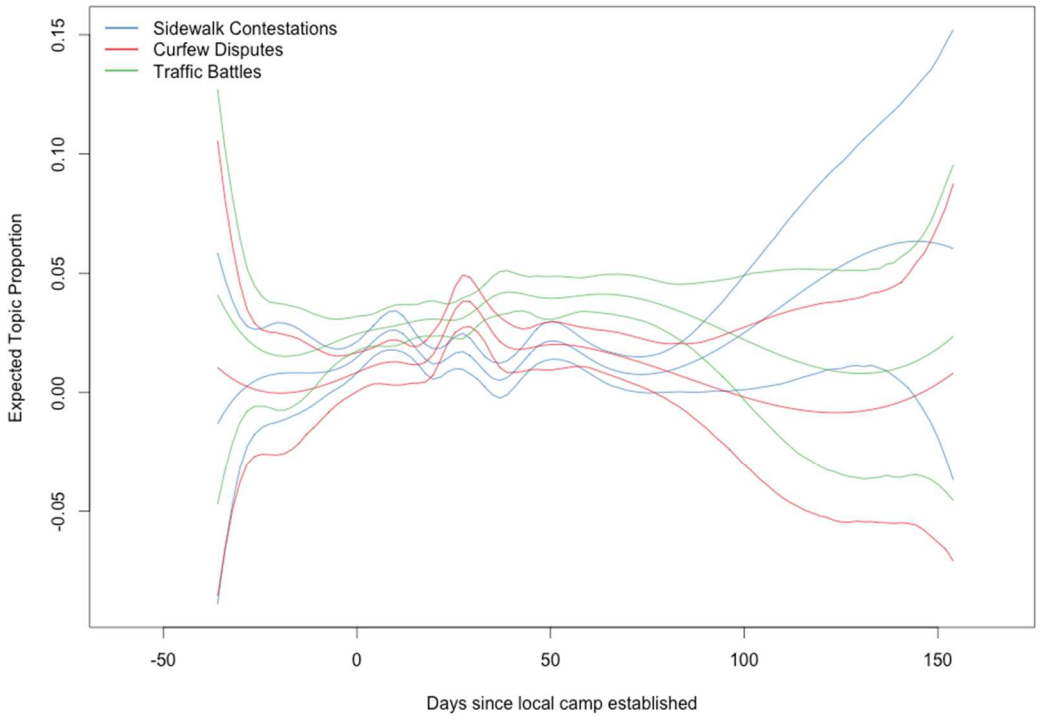
Neither these, nor other protester actions, occurred in a vacuum as the analyses of this chapter so far imply. It is somewhat disingenuous, in fact, to write of Occupy life courses without any reference to the police forces with which they contend. Indeed, a primary goal of this dissertation is to tease out the interactive relationships between police and protesters. This chapter and the next were designed to establish a baseline understanding of protester activity absent police-initiated events – an understanding to be corrected in later chapters. But, even before incorporating police data, LDA has refused to participate in the false abstraction of "protester activity" from "police activity." It is just as well.

Figure 4.5 shows that some of Occupy's contentious performances cannot be treated as protester-only behavior. These topics – related to police and protester contestation over sidewalk space, disputes about curfews, and battles over protester disruption of traffic – show that some protester activities, no matter how peaceful, consistently trigger police reactions. All of these topics involve disputable boundaries, "lines in the sand" that either police or protesters can decide to cross. Is the sidewalk public property open for civic activity including the confrontation of passersby, or is it a thoroughfare for foot traffic not to be blocked? Are curfews legal and enforceable, and under what circumstances are they worth enforcing? "Whose streets? Our streets!"

While other contentious performances center on winning or energizing new members, or targeting changes in the behavior of people (governments and banks) with the power to favorably respond to claims, activities identified by these topics involve direct confrontations with police about what constitutes reasonable disruptive action, about what constitutes civil disobedience vs. chaos. Observers and participants in the Occupy movement may recognize the pattern below. Sidewalk contestation, though resulting in few major clashes or arrests were a common way for police and protesters to attempt to assert their wills with one another. At no point, from the beginning to the end of the movement, did traffic disruption fail to garner a police response. As demonstrators in Seattle could attest (They tangled with police in the streets on days 10, 32, and 45 of their campaign.) police consistently cleared blocked intersections and roadways, whether through

warnings, arrests or pepper spray. On the other hand, curfew violations, for the early part of the movement, were generally not strictly enforced by police.

Figure 4.5 – Sidewalk Contestation, Curfew Disputes, Traffic Battles

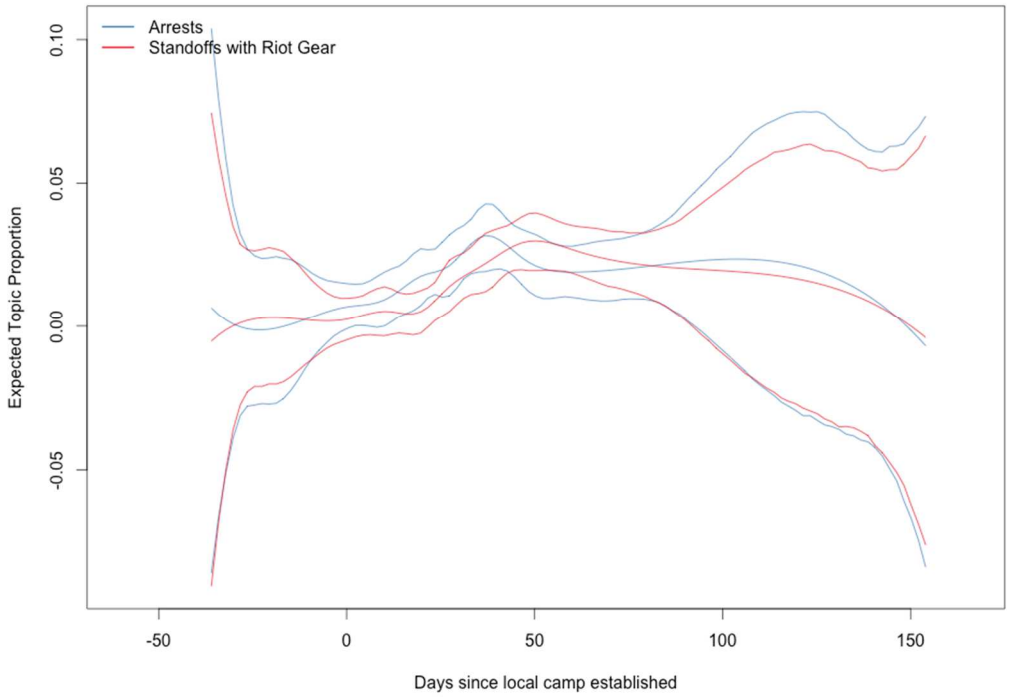


Note: Predicted prevalence of Sidewalk Contestation, Curfew Disputes, and Traffic Battles performances by day of encampment across the full corpus of text units describing protester-initiated contentious gatherings. All POS variables (number of liberals, number of power centers, and political instability) are held at their means.

While police in Cincinnati worked out a system whereby protesters lined up to receive nearly daily citations for curfew violations, many police departments behaved more like police in St. Louis. There, police had overlooked curfew violations for the first 4 weeks of the local encampment. Then, they began to threaten enforcement of curfew orders. And finally, in week 6, they forced the eviction of campers, citing the city's curfew ordinance. This rather common pattern is clear in Figures 4.5, 4.6, and 4.7. Curfew disputes hit their

peak 4 to 5 weeks into encampment. Figure 4.6 shows how those disputes resolved in most cases: with arrests in week 6 and greater use and threat of force by police in weeks 7 and 8.

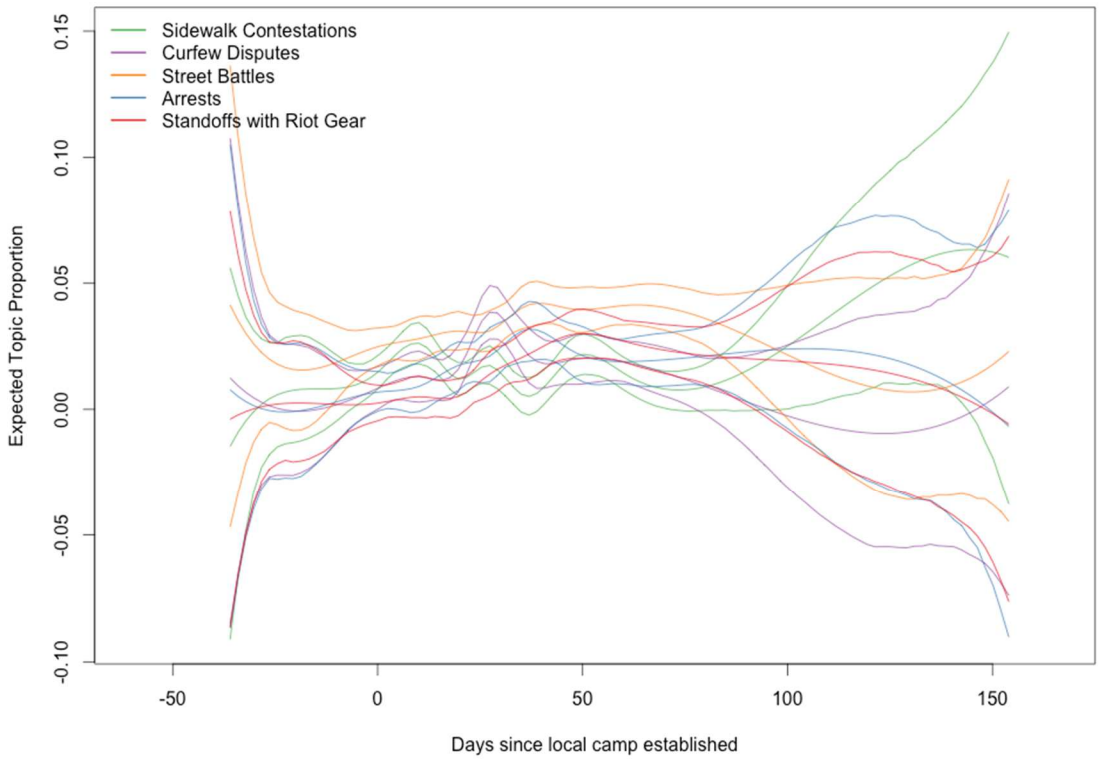
Figure 4.6 – Arrests, Standoffs with Riot Gear



Note: Predicted prevalence of Arrests and Standoffs with Riot Gear performances by day of encampment across the full corpus of text units describing protester-initiated contentious gatherings. All POS variables (number of liberals, number of power centers, and political instability) are held at their means

The escalation of police and protester mutual disdain is apparent in Figure 4.7, below. Police refuse to tolerate traffic disruption from Day 1. By the end of the first week of each encampment, rolling debates about sidewalks have begun. By the third weekend of the encampment, local police are warning protesters that curfew enforcement is a possibility. They show their willingness to enforce curfew orders in the fifth weekend, arresting protesters en masse in the sixth weekend. By the seventh weekend, many police departments deploy horse-mounted police and skirmish lines of riot police to show their resolve to prevent protesters from retaking the ground they had lost in weeks five and six.

Figure 4.7 – Policing Responses to Occupy Performances



Note: Predicted prevalence of Sidewalk Contestation, Curfew Disputes, Traffic Battles, Arrests and Standoffs with Riot Gear performances by day of encampment across the full corpus of text units describing protester-initiated contentious gatherings. All POS variables (number of liberals, number of power centers, and political instability) are held at their means

Discussion

These aggregate data, of course, cannot describe the details of the interactions between police and protesters across all US Occupy encampments. The data compiled for these

analyses do not even include data on events initiated by police (the subject of Chapter 6). But even from these preliminary analyses, a number of things are clear:

- The null hypotheses – that Occupy campaign activities were invariant over time (either individually or in the aggregate) – are incorrect.
- Occupy campaigns' contentious performances changed with the needs of the movement. The energy and participation of early demonstrations and rallies were channeled into building Encampments. As new members (like labor unions) joined the movement they were welcomed with Rallies (likely) designed to fortify their identification with the movement.
- The life courses of Occupy campaigns were, in some ways, significantly altered by police interventions. Energy was diverted into disputes about the physical and temporal boundaries enclosing the movement – sidewalks, curfews, and roadways.

These findings largely confirm Hypotheses 4.5 and support Hypothesis 6.x, later. But one detail of the life course hypotheses stated above appears to be at least partially incorrect. I had hypothesized that protesters would begin their activities with rather innocuous performances common to the traditional social movements repertoire (rallies, demonstrations and the like), and only escalate into more disruptive performances over time. This hypothesis appears to have been mistaken when it comes to performances including Traffic Battles with police. Though these actions did rise slightly over the course of Occupy campaigns, they appear to have been relatively common even in the beginning of the movement. The early use of street blocking defies the hypothesis that disruption was absent prior to any escalation of conflict with local police. The implications of this will be discussed further in Chapters 6 and 7.

Variations in Life Courses?

While the plots above show what Occupy campaigns were doing in the aggregate, one should not conclude that each campaign experienced an identical narrative.

There are many obvious ways that city variation might affect variation in these life courses. Large cities, for instance, house more, liberal, and more liberal, people than their smaller counterparts. In larger cities, therefore, the pool of potential Occupy activists and allies is significantly larger. Did this play a role in the prevalence of Occupy activities (or reports thereof)?

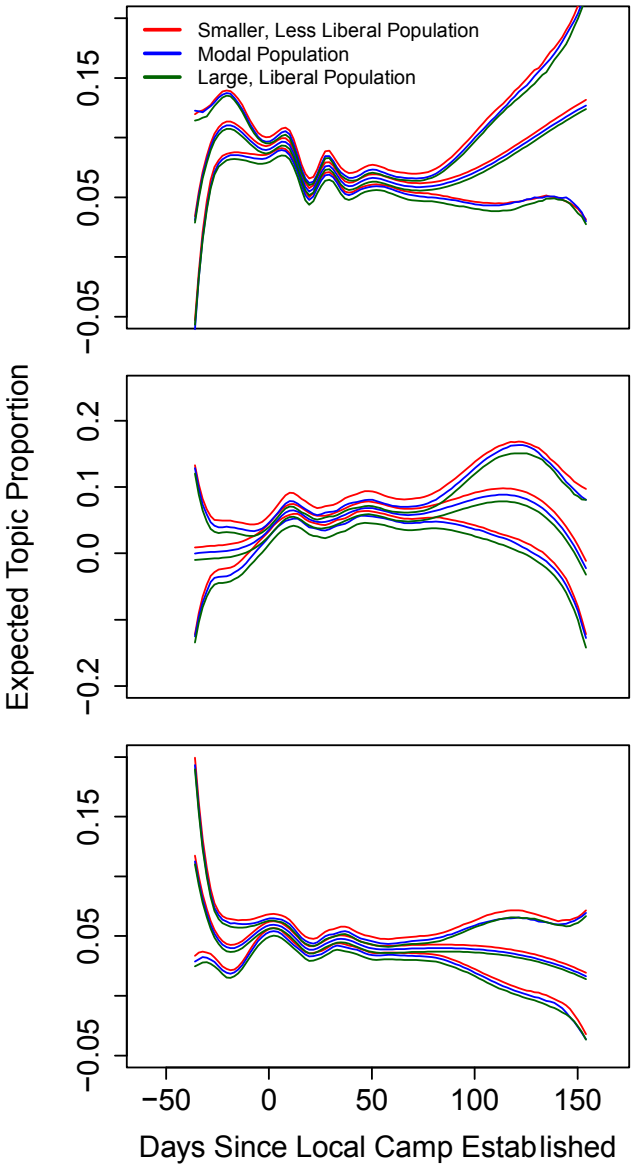
Figures 4.8 – 4.11 below, show topic/performance prevalence for towns and cities with large or small pools of potential Occupy activists. The pool of potential supporters is calculated by multiplying the population of the city or town by the percentage of voters who voted for Obama in the 2008 election. This pool measure, therefore, focuses observers' attention on the size of a city and its political liberality at the same time. (It also controls for reporting biases likely to emerge in differing media markets, as explained in Chapter 3.)

Figures 4.8 – 5.8 display a predicted performance prevalence γ across three different values of one β (e.g. β_1) for τ while holding the other β coefficients (e.g. β_2, β_3) at their means. All of these figures rely on the same general equation, below, and are described in further detail in the Methodological Appendix:

$$\gamma = \tau(x) + \beta_1(x) + \beta_2(x) + \beta_3(x) + \cdots \beta_n(x) + \varepsilon$$

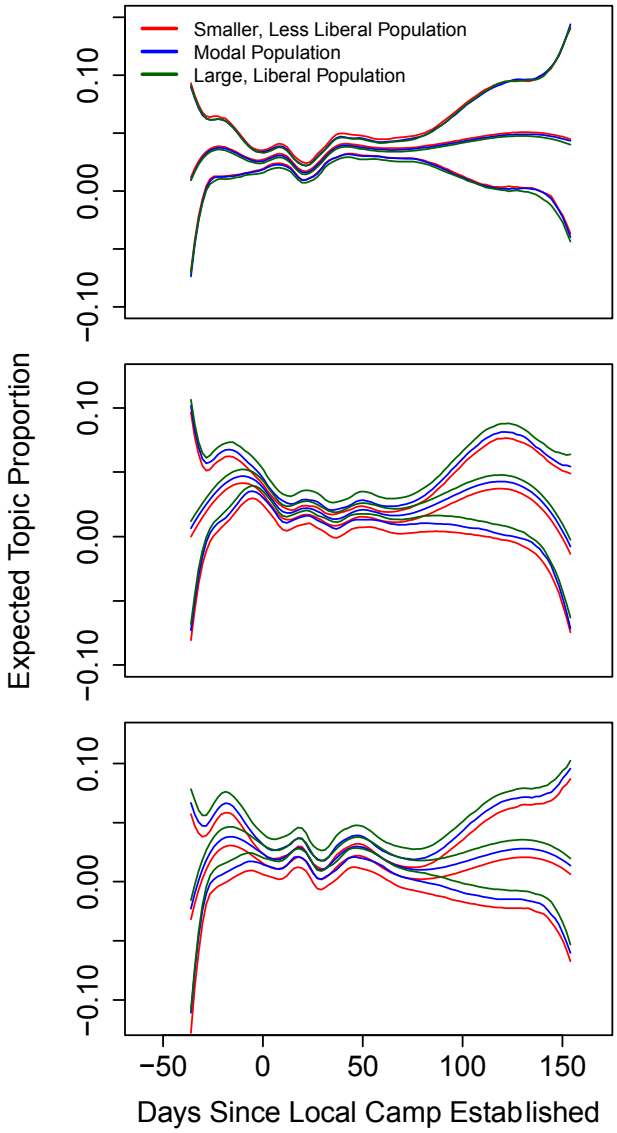
The 3 panels of Figure 4.8 show the prevalence of Weekend Gatherings, Encampment Activities, and Weekday Marches, respectively. Perhaps surprisingly, smaller, less liberal populations appear, at first glance, to have been more active than their counterparts in larger, more liberal cities. Note, however, that these differences are not statistically significant and may only represent that a higher proportion of local reporting focused on these activities. (Readers can assess whether the differences between blue and red lines are statistically significant by noting whether the middle line of one color is beyond the confidence interval line of another color.)

Figure 4.8 – Weekend Gatherings, Encampment Activities, Weekday Marches by Size of Liberal Population



Note: Panel 1 (Top), Panel 2 (Middle), and Panel 3 (Bottom) show the predicted prevalence of Weekend Gatherings, Encampment Activities, and Weekday Marches in the corpus for each day of the Occupy movement. Each panel displays three lines predicting topic prevalence for the highest, lowest, and middle value of the 'number of liberals' variable. See Methodological Appendix for equations used in estimation and prediction.

Figure 4.9 – Rallies, Demonstrations, and Labor Alliances by Size of Liberal Population

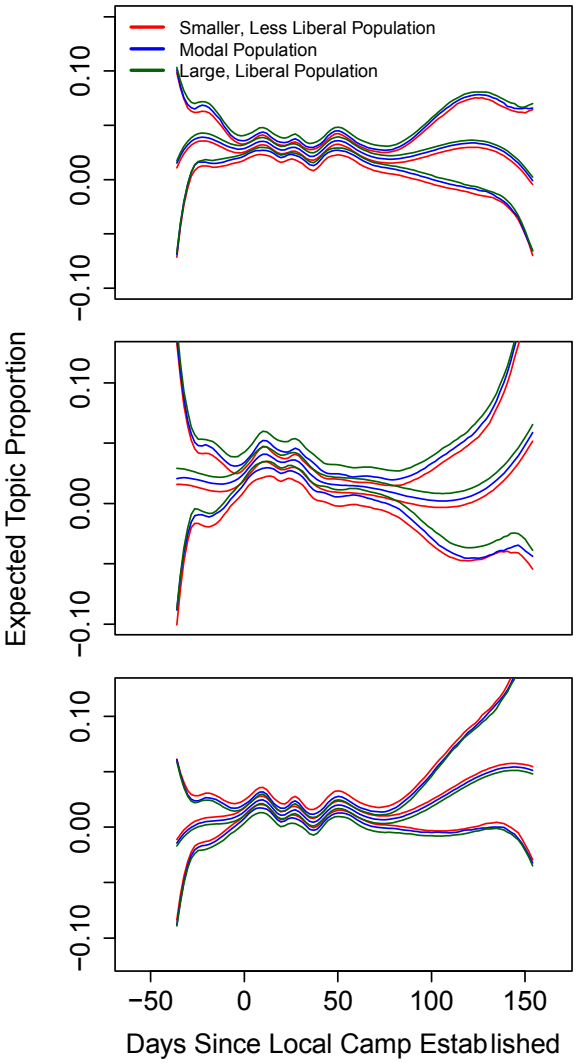


Note: Panel 1 (Top), Panel 2 (Middle), and Panel 3 (Bottom) show the predicted prevalence of Rallies, Demonstrations, and Labor Alliances in the corpus for each day of the Occupy movement. Each panel displays three lines predicting topic prevalence for the highest, lowest, and middle value of the 'number of liberals' variable. See Methodological Appendix for equations used in estimation and prediction.

That general pattern of activity does not appear to hold, however, when we observe the prevalence of contentious performances like Rallies and Demonstrations. Cities with smaller and larger pools of potential Occupy supporters engaged in a similar number of Rallies. However, cities with larger pools of political liberals appear to have been more likely to engage in Demonstrations and to have garnered more Union support during all of their activities. Larger cities appear to have engaged in more actions Targeting City Halls and Banks, as well (Panels 1 and 2 above). This difference is statistically significant in the case of Bank-targeting.[13]

[13] A note for interpreting the significance of any apparent differences: whenever a topic prevalence estimate line (the middle line) of one color falls outside the confidence interval (either of the two outer lines) of the other color, the model predicts that the differences between estimates of the two values of the covariate are statistically significant.

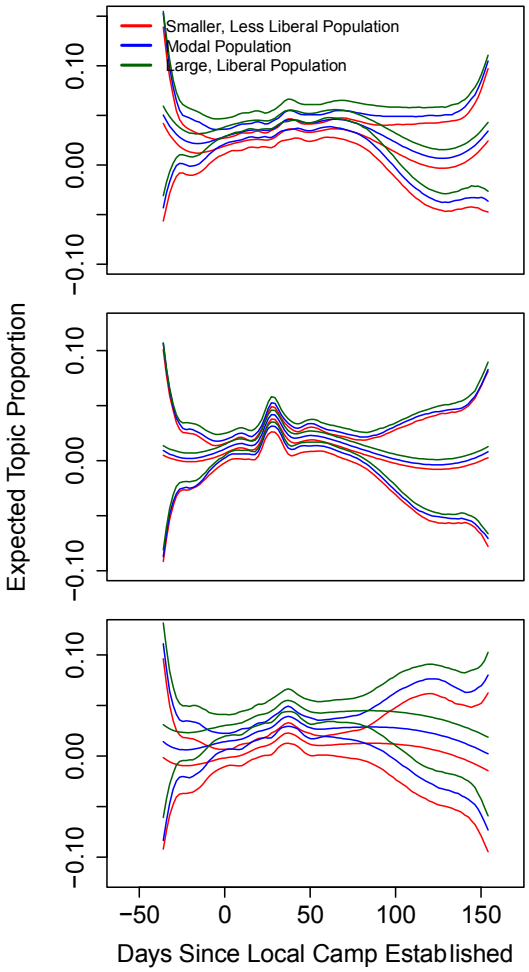
Figure 4.10 – City Hall Targeting, Bank Targeting, and Sidewalk Contestation by Size of Liberal Population



Note: Panel 1 (Top), Panel 2 (Middle), and Panel 3 (Bottom) show the predicted prevalence of City Hall Targeting, Bank Targeting, and Sidewalk Contestation in the corpus for each day of the Occupy movement. Each panel displays three lines predicting topic prevalence for the highest, lowest, and middle value of the 'number of liberals' variable. See Methodological Appendix for equations used in estimation and prediction.

Campaigns in large and small, liberal and less liberal, cities appear to have differed, too, in the amount of police resistance they experienced. Larger cities' campaigns appear to have included more Traffic Blocking actions arousing the ire of police, and more Curfew Disputes resulting in statistically significantly more Arrests, as well. Explaining these differences and other variations in Occupy campaigns will be the task of Chapter 5.

Figure 4.11 – Traffic Battles, Curfew Disputes, Arrests by Size of Liberal Population



Note: Panel 1 (Top), Panel 2 (Middle), and Panel 3 (Bottom) show the predicted prevalence of Traffic Battles, Curfew Disputes, and Arrests in the corpus for each day of the Occupy movement. Each panel displays three lines predicting topic prevalence for the highest, lowest, and middle value of the 'number of liberals' variable. See Methodological Appendix for equations used in estimation and prediction.

Chapter 5: The Correlates of Contentious Performances and Campaign Life Courses: Political Opportunity Structures

Findings from Chapter 4 suggest that Occupy campaigns may have engaged in a similar sequence of activities during the 'action' stage of the Occupy movement. That is, they may have experienced a similar life course during the 'action' stage. However, an initial investigation of the effect of city population size and liberality on the prevalence of contentious performances suggests that not all Occupy campaigns were the same. A larger proportion of campaign activities in small cities and towns centered on Encampment activities, while a greater proportion of campaigns' efforts in larger cities was dedicated to Rallies and Demonstrations. Larger cities spent more time targeting banks, too, and engaged in more disruptive performances arousing more, and firmer, police responses.

It is important to account for these differences because the ultimate goal of this dissertation is to determine which factors – among features of cities, features of police departments, and features of Occupy campaigns – explain the policing of protest campaigns (and when and under what circumstances.) Those questions can only be answered if it is clear either (a) that Occupy campaigns were very similar and/or (b) that they differed only in ways that can be measured and controlled for statistically. This chapter aims to more thoroughly assess the comparability of Occupy campaigns to determine if and how they may be used as a basis for understanding police control performances in Chapter 6.

Students of the sociological literature on social movements will not be surprised by the finding (from Chapter 4) that the population of potential movement supporters affected the prevalence of Occupy campaign activities. Even when social movement campaigns are animated by very similar grievances, make similar claims, and draw on similar repertoires of contention, their activities are likely to vary based on the social and political environments in which they emerge. As Shultziner notes, "structural factors are often crucial at [the action] stage" of a movement. A thorough literature (reviewed in Chapter 2) attests to the influence of structural variables on movement action and outcomes.

Besides being limited or enabled by a pool a potential supporters, campaign actions seeking to mobilize support among elites, polity members, challengers, and outsiders depend on the composition of that milieu, the political cultures that define their identities, and the political structures which produce and constrain their powers. Taken together, these "features of regimes and institutions that facilitate or inhibit a political actor's collective action and… changes in those features" are referred to as 'political opportunity structures' (Tilly and Tarrow 2006, 49).Tilly defines six dimensions of political opportunity structure:

> (a) the multiplicity of independent centers of power within the regime,
>
> (b) the openness of the regime to new actors,
>
> (c) the instability of current political alignments,
>
> (d) the availability of influential allies or supporters,
>
> (e) the extent to which the regime represses or facilitates collective claim-making, and (f) decisive changes in (a) to (e).

There has been no small amount of debate about the conceptualization of "political opportunity structures." (See, for instance, Kriesi 1995; Koopmans 1999.) A range of scholars, though have moved forward, attempting to assess the impact of POS on movement activities and outcomes. Kitschelt (1986) and Kriesi (1995) have argued, alongside Tilly that governments featuring more centers of power (across executive, legislative, and judicial functions) feature more sources of potential movement allies, encouraging movement activity. Others, however, suggest that these government divisions also imply more veto points, discouraging movement activity (Huber, Ragin and Stephens 1993; Skocpol 1992; Amenta and Young 1999; Amenta and Caren 2004, 472).

There are fewer doubts about the encouraging effects openness of regime and instability of political alignments have on movements. Piven and Cloward (1977, 31-2) have found that electoral instability seems to be an important factor in the efficacy of even high-impact civil disruptions like industrial labor strikes. And Tilly (1978, 213-14) found that alliances among governments and challengers were most likely during close elections. Multiple

authors, too, have found that center-left parties sometimes ignore leftist movements unless elections are near (Pizzorno 1978; Pizzorno 1981; della Porta 2001).

Since movement actions rise and fall with levels of support, the availability of allies and supporters will obviously impact their activity. While no one doubts that mass movements' attempts to display WUNC (worthiness, unity, numbers, and commitment) (Tilly 2005, 53) depend on the size of the pool of supporters from which they draw, it is less clear whether the local ecosystem of existing social movement organizations consistently affects the activities of social movement campaigns. Some movements, like the U.S. Civil Rights Movement of the 1960s, depended greatly on the organizational support and networks of churches, foundations, and anti-poverty programs. (McAdam 1982; Morris 1984; della Porta and Diani 2006). Many movements of the left, too, have benefited from the solidarity of organized labor (Clawson, 2003). But, a rich SMO environment can also reduce movement activity by alleviating or channeling the grievances that animate a movement (Jenkins and Leicht 1997: 378–9). Segura-Diaz (2015), in a study using this dissertation's data, even showed that 'Occupy the Hood' campaigns, offshoots of the broader Occupy movement, were most likely to develop in cities that lacked the social service, and social movement, organizations that would typically address the concerns of low-income minority communities.

This dissertation explores the impact of these elements of POS on the range of Occupy campaign activities identified in Chapter 4. I will argue (more thoroughly in Chapters 6 and 7) that element (e) is a complex set of dynamic variables to explain, not a relatively stable *feature* of a regime. The remaining elements of POS a–d, however, I operationalize through a number of variables describing local cities'/towns' political environments.

These operationalizations are designed for an analysis seeking to describe and explain the deployment of contentious activities and performances at a more granular level of geographic resolution than Tilly's data would allow. Tilly deployed the concept of political opportunity structures quantitatively in *Regimes and Repertoires* (2006, Chapter4) with an index of 'regime capacity' and a freedom house 'democracy scale.' But these measures neither attempted to capture political opportunity structure at a local, city/town level, nor captured every element of his rather expansive definition.

Rather than relying on broad-scale data describing national regimes, I operationalize each element of POS listed above using the following variables:

(a) 'Number of Power Centers' – Across the U.S., three city government types predominate: Mayor-Council, Council-Manager, and Commission. The Mayor-Council type of government features a relatively strong executive and relatively week City Council. In terms of POS, it features the fewest 'independent centers of power." The Council-Manger government type features more powerful council members with their own center of power. And the Commission government type features a number of committees or commissions, each with legislative and executive authority over a specific domain of municipal governance. 'Government type' data have been collected for every city in this dissertation's dataset and arranged on a scale from 1-3, ranging from fewest to most independent centers of power. If a city is also the seat of State power, one point is added to this score. Since no State Capitals are run via Commission, all cities' Government type scores fell between 1 and 3.

(b) 'Openness to New Actors'– U.S. city governments are most 'open to new actors' immediately prior to elections. And in the run-up to elections, politicians are most likely to alter their behavior to please their constituents. To measure local government openness to new actors, I subtracted each Occupy campaign's start date from the date of the next upcoming election. If the next election was within 3 months, the city's 'openness to new actors' score was tallied as a 3; within 3-6 months, a 2, within a year, a 1; beyond a year, 0.

(c) 'Instability of Political Alignments'– Political alignments are least secure when they are just forming, just after elections. Using the same procedure as detailed in (b), I measured the proximity of an Occupy campaign's start date to the latest city election. In the analyses that follow I have combined (b) and (c), using a simple average, as a measure of Political Instability.

(d) 'Availability of Allies and Supporters' – Given that the Occupy movements were largely conceived and perceived as movements of the left, an Occupy campaign's base of potential support, element (d) of POS, can be operationalized as # of people who voted for Obama in the 2008 election, a number derived by multiplying Obama's vote percentage by the city's/town's population according to the 2010 U.S. Census.

(e) 'The Extent of State Repression' – (This element of political opportunity structure is a dependent variable incorporated into the analyses and discussion of Chapters 6 and 7)
(f) 'Decisive Changes in (a)-(e)' – (Neither (a) nor (d) would have changed at all during the Fall of 2011. Elements (b) and (c) may have changed somewhat, but in ways that are normalized by the foregoing analyses.

Equipped with variable data describing the elements of POS for all 184 U.S. cities in which Occupy campaigns occurred, this chapter asks if and how those POSs affected the enactment of contentious performances by Occupy campaigns. These questions are asked and answered using a variant of the LDA/topic modeling technique described in detail in Chapter 3, *structural* topic modeling. Structural topic modeling begins with the same algorithm used by LDA/topic modeling. It first produces 'topic' output, *K* lists (where *K* is the number of 'topics' defined by researchers), each including a different ordering of the full set of unique terms included in the researchers' text corpus. The term lists are ordered by the likelihood that the terms appear alongside one another in the same text unit.

Since the terms of this dissertation's text corpus describe *activities* of protesters and police during *events* that are conterminous with the text units being modeled by LDA, the 'topic' lists of this dissertation may be said to identify *contentious performances*: sets of action that reliably cohere as elements (even sometimes the predominate element of) *contentious gatherings.*

Structural topic modeling uses well-known regression techniques to regress structural variables linked to the text units (variables like the size and political form of the cities in which the news documents were produced) on topic prevalence for those text units. The prevalence of performances in contentious gatherings (described by any or all text unit/s), therefore, can be modeled as a function of contentious gathering-level variables like the 'city' in which a contentious gathering described in the text occurred, the date on which it occurred, the 'population' of the city in which it occurred, the 'government type' of that city, the POS variables listed as (a) through (d) above for that city, and so on.

Whether or not the city-level POS variables affect protesters' performances is an open question. Tilly, in his words, has "shown amply that contentious repertoires differ

dramatically from one type of regime to another." But Tilly was comparing national regimes at the corners of his 2X2 table spanning high- and low-capacity regimes, and democratic/nondemocratic regimes. No one would argue that American cities demonstrate as much range on these dimensions as states like Somalia, Uganda, South Africa, Venezuela, India, Norway, and the U.S. Other authors have estimated the effects of U.S. (subnational) State-level political opportunities on social movement activities (Amenta and Zylan 1991). Will interesting correlations between city 'regime' POS variables and performance prevalence emerge even though the differences in POSs among American cities are relatively small?

I make the following hypotheses in line with Tilly's general theory:

Hypothesis 5.1: In cities featuring more independent centers of power, Occupy campaigns will be more active.

Hypothesis 5.2: In cities that have just experienced or will soon experience an election, campaigns will be more active, reflecting the general political activity concomitant with local political realignments.

Hypothesis 5.3: In cities that have just experienced or will soon experience an election, elites will be more solicitous of Occupy campaigns, and as a consequence, protesters' performances will be less disruptive.

Hypothesis 5.4: In cities where more Obama voters live, campaigns will be more active

Hypothesis 5.5: In cities where more Obama voters live, campaigns will feature more performances designed for mass crowds, like marches and demonstrations.

Model Results

Readers will recall from Chapter 4 that estimates of performance prevalence are generated through the following regression equation:

$$\gamma = \tau(x) + \beta_1(x) + \beta_2(x) + \beta_3(x) + \cdots \beta_n(x) + \varepsilon$$

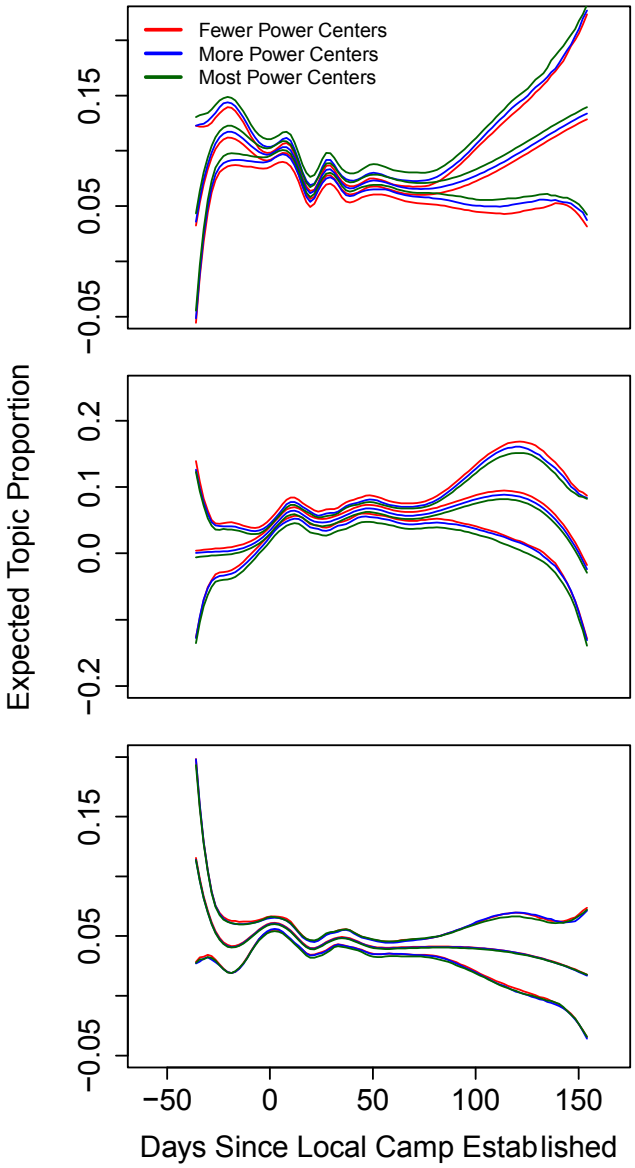
Figures 5.1 – 5.8 display a predicted performance prevalence γ across three different values of one β (e.g. β_1) for τ while holding the other β coefficients (e.g. β_2, β_3) at their means. (See the Methodological Appendix for more details.)

The plots in Figures 5.1 -5.4, similar in form to those ending Chapter 4, show the prevalence of topics/performances since the first day of local encampment. In these plots, red lines indicate towns and cities with relatively few centers of power, while blue lines represent cities with more centers of power.[14] If Tilly's theory holds even for rather small differences in regime composition (The range of American cities' diversity is significantly narrower than the range for states.), protesters activity should be more prevalent in cities represented by blue lines, which should appear higher on the y-axis than red lines for most of the activities identified by LDA. A note for interpreting the significance of any apparent differences: whenever a topic prevalence estimate line (the middle line) of one color falls outside the confidence interval (either of the two outer lines) of the other color, the model predicts that the differences between estimates of the two values of the covariate are statistically significant.

The first panel of Figure5.1 below, shows that cities with more power centers experienced statistically significantly more Weekend Gathering activities than cities with relatively fewer power centers (at least during the most active portions of the Occupy movement). Panel 2, on the other hand, shows that cities with fewer power centers experienced more Encampment Activities than cities with more power centers (or at least that these activities were more prominent in reporting about these local campaigns), but that the differences across these categories of city/town are not statistically significant. Panel three shows that there was virtually no difference in the prevalence of Weekday Marching activities (or reports thereof) across cities with different numbers of power centers.

[14] See Methodological Appendix for variables' assignments to cities.

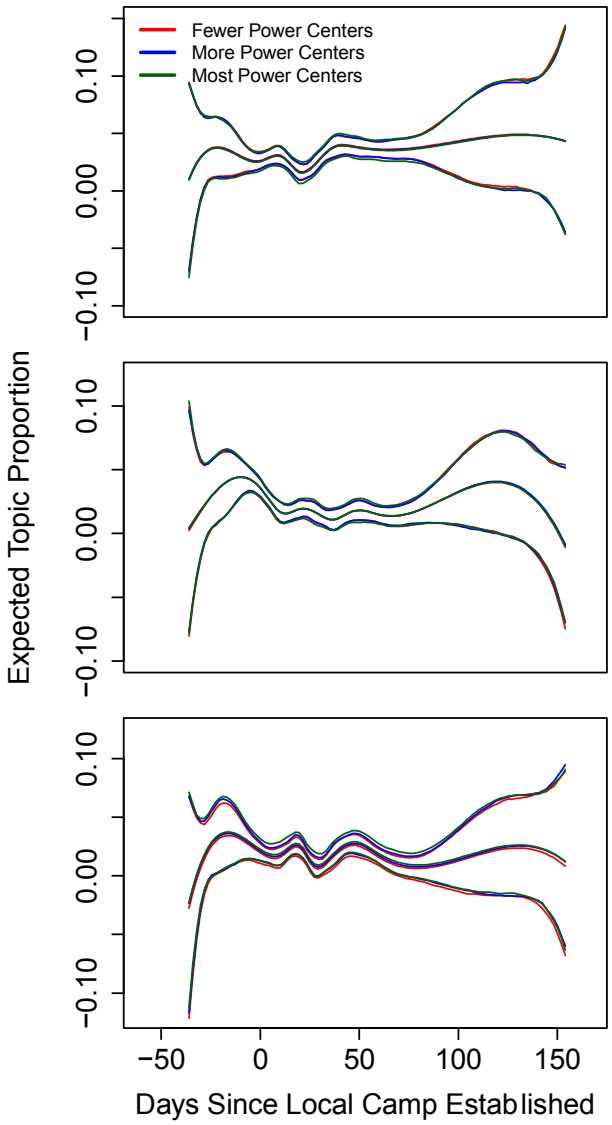
Figure 5.1 – Weekend Gatherings, Encampment Activities, Weekday Marches by Number of Power Centers



Note: Panel 1 (Top), Panel 2 (Middle), and Panel 3 (Bottom) show the predicted prevalence of Weekend Gatherings, Encampment Activities, Weekday Marches in the corpus for each day of the Occupy movement. Each panel displays three lines predicting topic prevalence for the highest, lowest, and middle value of the 'number of power centers' variable. See Methodological Appendix for equations used in estimation and prediction.

Figure 5.2 below, shows that number of power centers, likewise, had little to no effect on the prevalence of Rallies, Demonstration, or Labor Alliances.

Figure 5.2 – Rallies, Demonstrations, and Labor Alliances by Number of Power Centers

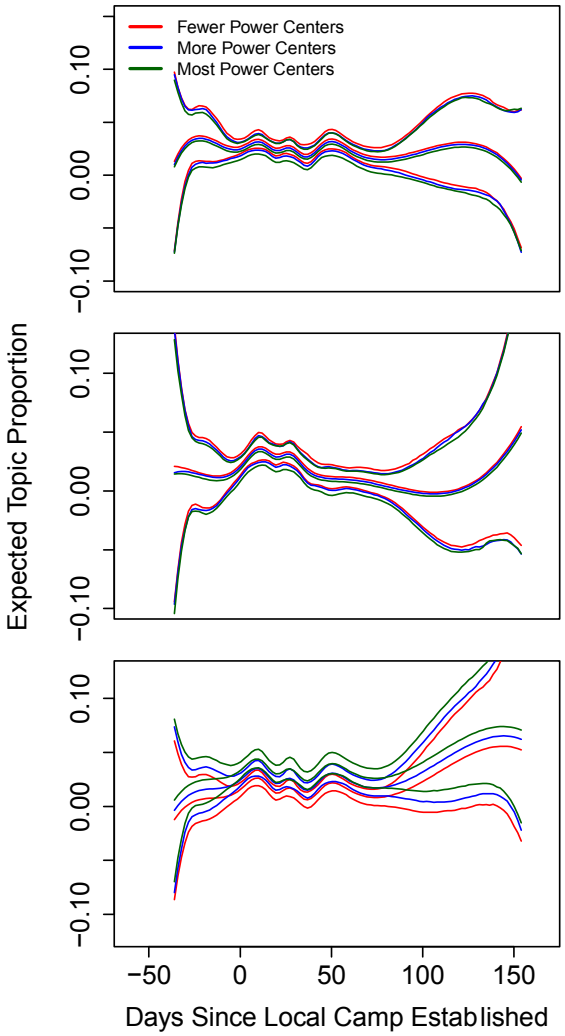


Note: Panel 1 (Top), Panel 2 (Middle), and Panel 3 (Bottom) show the predicted prevalence of Rallies, Demonstrations, and Labor Alliances in the corpus for each day of the Occupy movement. Each panel displays three lines predicting topic prevalence

for the highest, lowest, and middle value of the 'number of power centers' variable. See Methodological Appendix for equations used in estimation and prediction.

The number of power centers, likewise, had little effect on the prevalence of reporting on protesters' targeting of City Halls or Banks. However, whether or not such targeting differed across cities led by Strong Mayors versus cities led by Commissions or Councils, the former cities did seem to experience more contestation over the sidewalks on which that targeting often took place.

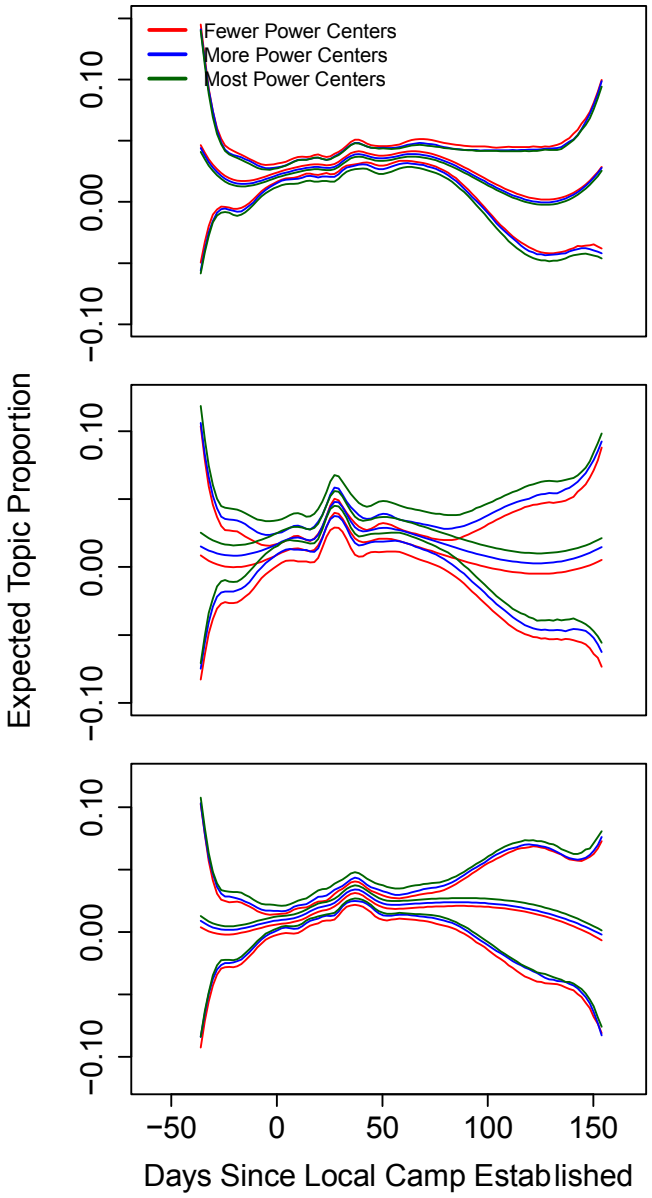
Figure 5.3 – City Hall Targeting, Bank Targeting, and Sidewalk Contestation by Number of Power Centers



Note: Panel 1 (Top), Panel 2 (Middle), and Panel 3 (Bottom) show the predicted prevalence of City Hall Targeting, Bank Targeting, and Sidewalk Contestation in the corpus for each day of the Occupy movement. Each panel displays three lines predicting topic prevalence for the highest, lowest, and middle value of the 'number of power centers' variable. See Methodological Appendix for equations used in estimation and prediction.

Cities with more centralized authority, too, seemed to experience more curfew disputes (Panel 2 in Figure 5.4 below) than their counterpart cities with more distributed leadership. The reasons for this will be explored in more detail in Chapter 6.

Figure 5.4 – Traffic Battles, Curfew Disputes, and Arrests by Number of Power Centers

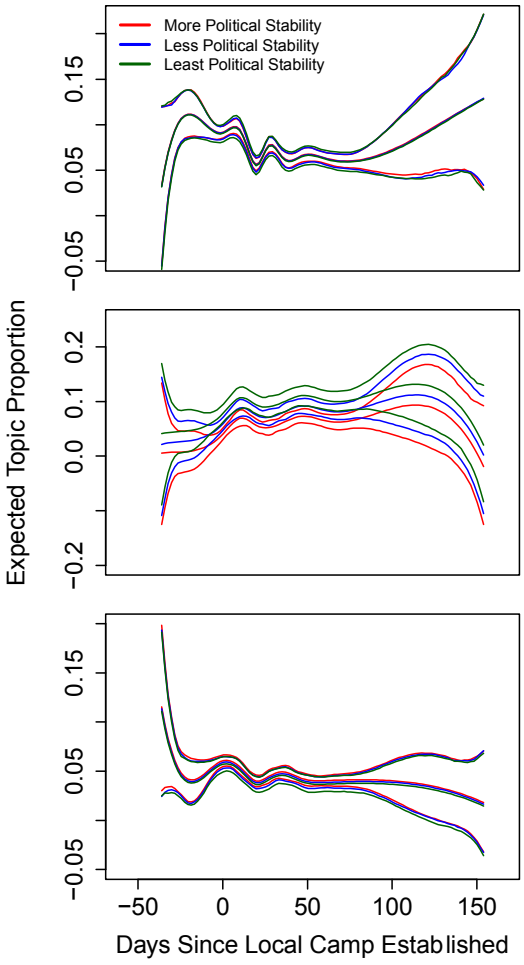


Note: Panel 1 (Top), Panel 2 (Middle), and Panel 3 (Bottom) show the predicted prevalence of Traffic Battles, Curfew Disputes, and Arrests in the corpus for each day of the Occupy movement. Each panel displays three lines predicting topic prevalence for the highest, lowest, and middle value of the 'number of power centers' variable. See Methodological Appendix for equations used in estimation and prediction.

Political Stability and its Effect on Occupy Activities

Politicians are most responsive to their constituencies immediately prior to elections. And they are often most open to change in moments of transitioning political alignments. Tilly and others argue that social movement organizers understand this and attempt to take advantage of these windows of opportunity to influence political and policy outcomes. If their theories are correct, we should expect cities experiencing greater electoral and political instability to experience more active Occupy campaigns. Figures 5.5 – 5.8, below, offer some support for this general theory.

Figure 5.5 – Weekend Gatherings, Encampment Activities, Weekday Marches by Political Instability

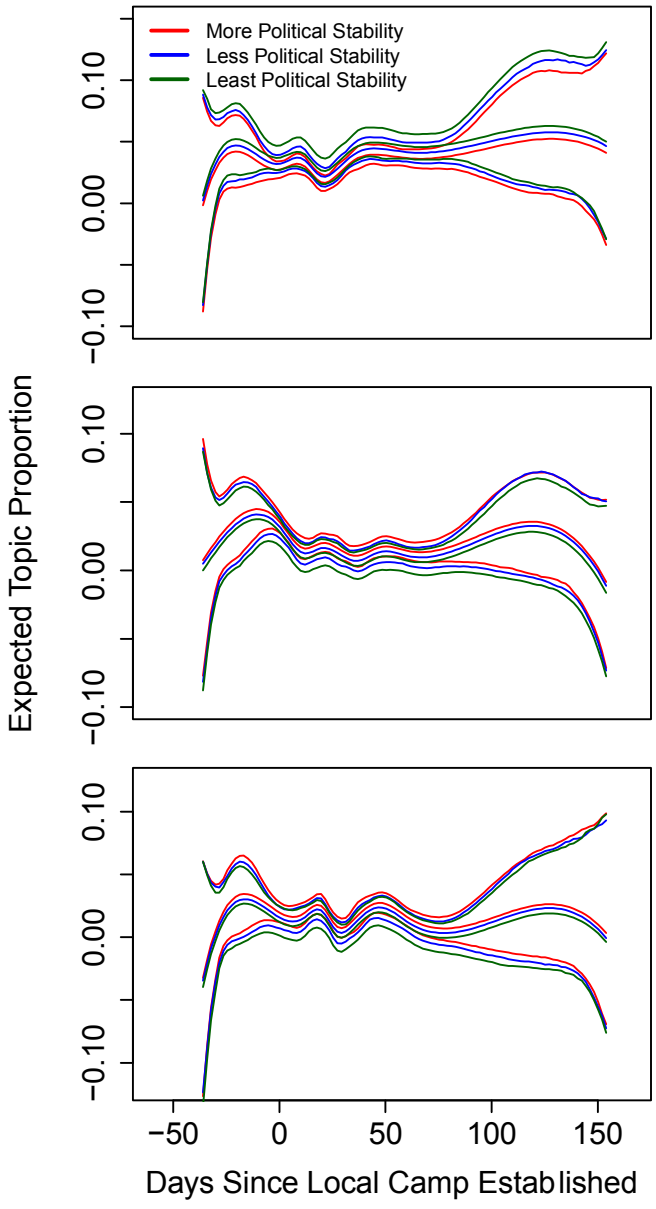


Note: Panel 1 (Top), Panel 2 (Middle), and Panel 3 (Bottom) show the predicted prevalence of Weekend Gatherings, Encampment Activities, and Weekday Marches in the corpus for each day of the Occupy movement. Each panel displays three lines predicting topic prevalence for the highest, lowest, and middle value of the 'political instability' variable. See Methodological Appendix for equations used in estimation and prediction.

Panel 2 of Figure 5.5, above, for instance, shows that Encampments appeared to be more active in cities that experienced an Election Day around the time of the Occupy movement. (Also, note that lines are jagged because they are displaying two values of the 'political instability' variable at once. There was no way to reduce the visualization of this variable to two smooth lines without also unduly reducing the information in the variable during model.)

According to Figure 5.6, proximity to an election had relatively less influence on local Occupy campaigns' engagement in Rallies, Demonstrations, or Labor Alliances.

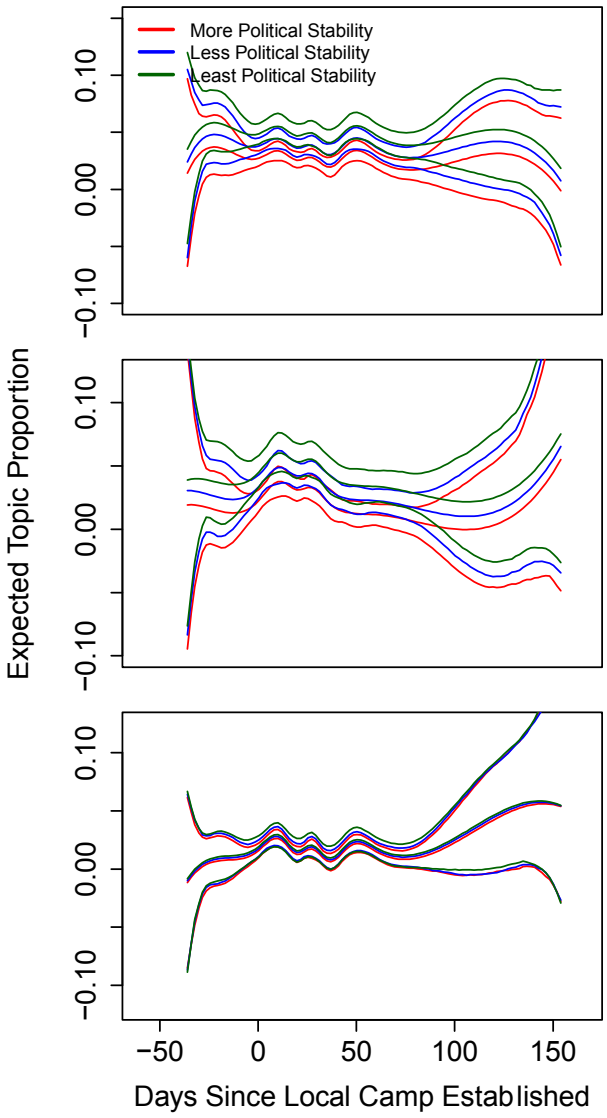
Figure 5.6 – Rallies, Demonstrations, and Labor Alliances by Political Instability



Note: Panel 1 (Top), Panel 2 (Middle), and Panel 3 (Bottom) show the predicted prevalence of Rallies, Demonstrations, and Labor Alliances in the corpus for each day of the Occupy movement. Each panel displays three lines predicting topic prevalence for the highest, lowest, and middle value of the 'political instability' variable. See Methodological Appendix for equations used in estimation and prediction.

However, as Figure 5.7 show, Occupiers did seem to seize on political instability by targeting City Halls and Banks with more of their activities. Electoral insecurity, however, seems not to have had an impact on the extent to which these and other actions resulted in disputes around sidewalk territory.

Figure 5.7 – City Hall Targeting, Bank Targeting, and Sidewalk Contestation

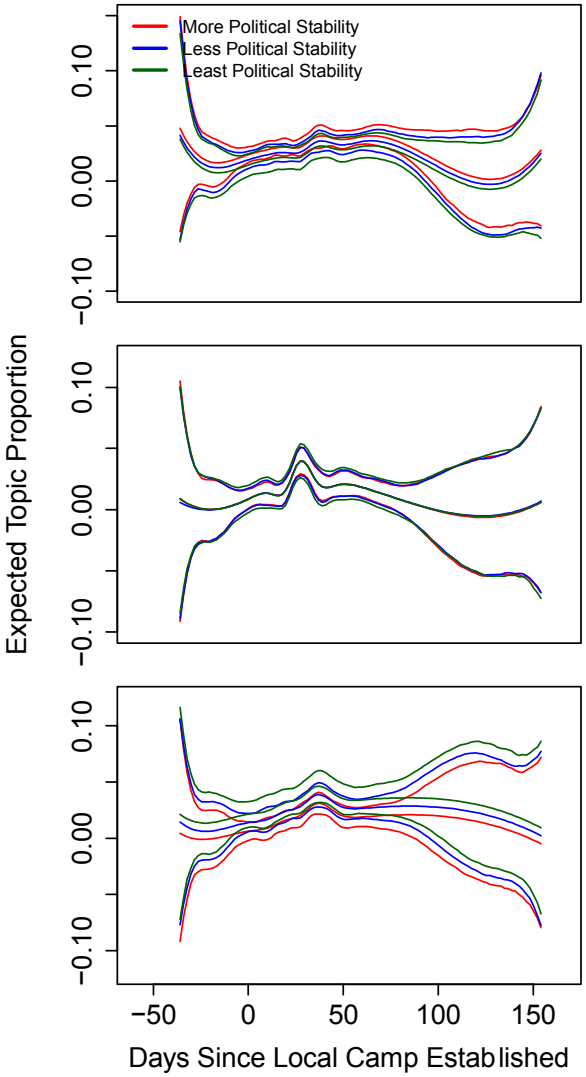


Note: Panel 1 (Top), Panel 2 (Middle), and Panel 3 (Bottom) show the predicted prevalence of City Hall Targeting, Bank Targeting, and Sidewalk Contestation in the corpus for each day of the Occupy movement. Each panel displays three lines

predicting topic prevalence for the highest, lowest, and middle value of the 'political instability' variable. See Methodological Appendix for equations used in estimation and prediction

Instead, perhaps, as Panel 1 of Figure 5.8 suggests, Occupy campaigns in cities experiencing electoral instability used City Hall and Bank targeting actions in place of traffic disruption activities. These different choices of targets could have resulted in the greater prevalence of arrests as shown in Panel 3.

Figure 5.8 – Traffic Battles, Curfew Disputes, and Arrests by Political Instability



Note: Panel 1 (Top), Panel 2 (Middle), and Panel 3 (Bottom) show the predicted prevalence of Traffic Battles, Curfew Disputes, and Arrests in the corpus for each day of the Occupy movement. Each panel displays three lines predicting topic prevalence for the highest, lowest, and middle value of the 'political instability' variable. See Methodological Appendix for equations used in estimation and prediction.

The models above show that political opportunity structures do seem to have shaped local Occupy campaigns' choices of targets and activities (or at least the reporting thereof) in some (statistically) significant ways.

Hypotheses 5.2, 5.3, 5.4 and 5.5 all won some support. Scholars arguing that more centers of power do not make for greater activity seem to have won support as well: Hypothesis 5.1 was weakened by these findings.[15]

However, it is not clear from these models whether differently situated Occupy campaigns simply emphasized some contentious performances more than others, or if they experienced fundamentally different life courses altogether. Did differing political opportunity structures significantly re-order the sequences of gatherings and performances in which local Occupy campaigns engaged?

Political Opportunity Structures and Occupy Campaign Life Courses

[15] *Hypothesis 5.1*: In cities featuring more independent centers of power, Occupy campaigns will be more active. *Hypothesis 5.2*: In cities that have just experienced or will soon experience an election, campaigns will be more active, reflecting the general political activity concomitant with local political realignments. *Hypothesis 5.3*: In cities that have just experienced or will soon experience an election, elites will be more solicitous of Occupy campaigns, and as a consequence, protesters' performances will be less disruptive. *Hypothesis 5.4*: In cities where more Obama voters live, campaigns will be more active. *Hypothesis 5.5*: In cities where more Obama voters live, campaigns will feature more performances designed for mass crowds, like marches and demonstrations.

This final section of Chapter 5, presents plots of models able to answer this more difficult and nuanced question. Starting with Figures 5.9 - 5.12 showing the effect that smaller or larger pools of Obama supporters have on topic prevalence, these plots are able to show how political opportunity structures affect not just the quantity of Occupy related performances and gatherings (and/or their reporting), but also the timing of those performances and gatherings.

The models used to investigate more sophisticated hypotheses about the prevalence *and* timing of contentious performances require one key change, the introduction of a term interacting an independent variable with time: $\beta_1 \times \tau(x)$.

$$\gamma = \tau(x) + \beta_1 \times \tau(x) + \beta_2(x) + \beta_3(x) + \cdots \beta_n(x) + \varepsilon$$

The remaining figures of this chapter, and this dissertation, use the above equation to estimate $\beta_1, \beta_2, \beta_3, \ldots \beta_n$ then predicted values of γ across three different values of one β for τ while holding the other β coefficients (e.g. β_2, β_3) at their means. And since these models include the interaction term $\beta_1 \times \tau$, predictions of γ are not simply raised or lowered vertically as in Figures 4.8 – 5.8. The shapes of the γ prediction lines for each value of β across τ move with the data through time.

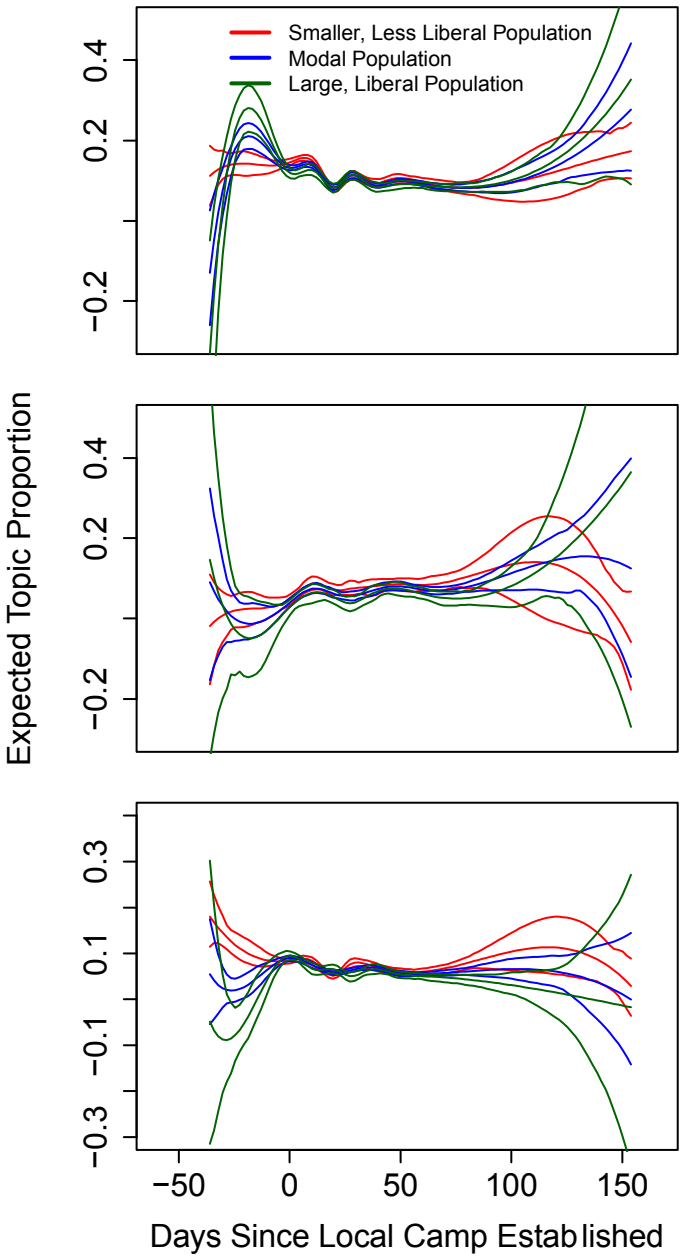
Pools of Support and Occupy Campaign Life Courses

Hypotheses 5.4 and 5.5

Panels 1 and 2 of Figure 5.9, below, show that cities with small, medium, and large pools of potential Occupy supporters engaged in similar levels of Weekend Gatherings and Encampment Activities throughout the course of their campaigns. Panel 3, however, reveals what Figure 4.8 in Chapter 4 could not, that Weekday marching activities in cities with larger pools of support started earlier and peaked three times before the seventh week of local campaigns (while such activities only peaked twice in cities with a smaller pool of potential supporters). Did the larger pool of supporters allow for quicker

mobilization and more frequent marches? Perhaps. Such questions will be discussed further in Chapter 7 in light of more complete data.

Figure 5.9 – Weekend Gatherings, Encampment Activities, Weekday Marches by Size of Liberal Population Interacted with Time

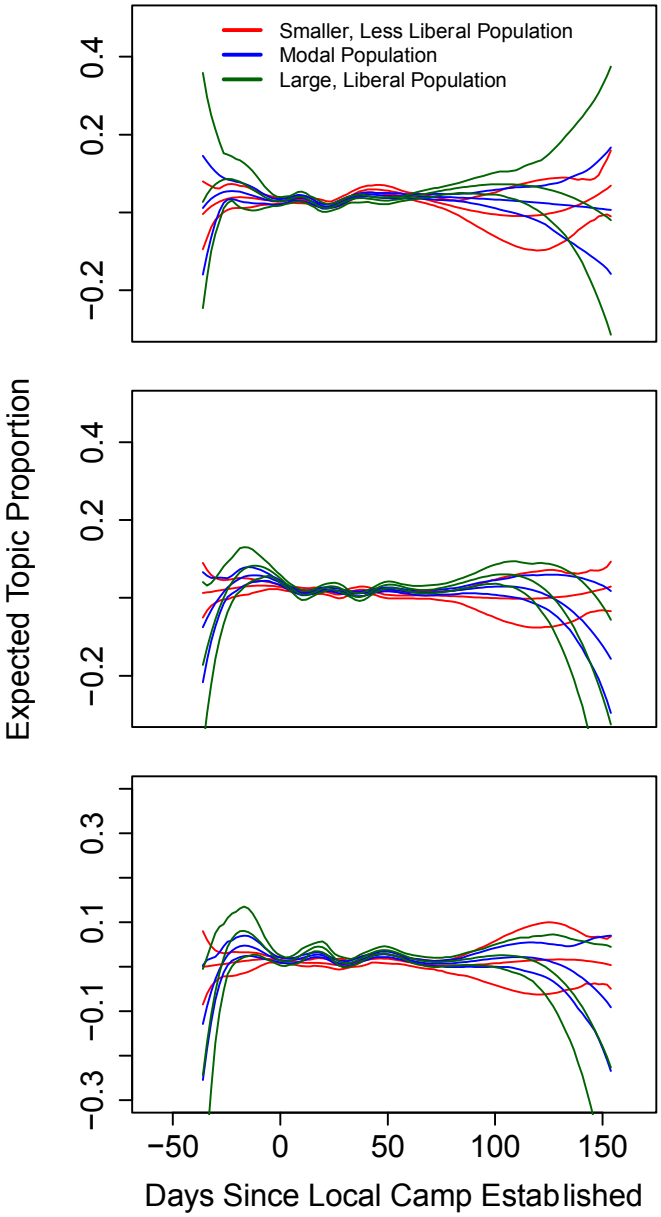


Note: Panel 1 (Top), Panel 2 (Middle), and Panel 3 (Bottom) show the predicted prevalence of Weekend Gatherings, Encampment Activities, Weekday Marches in the

corpus for each day of the Occupy movement. Each panel displays three lines predicting topic prevalence for the highest, lowest, and middle value of the 'number of liberals' variable interacted with time. See Methodological Appendix for equations used in estimation and prediction.

Figures 5.10, below, show that the rhythm of Rallies and Demonstrations also differed across cities depending on how many potential supporters local Occupy campaigns could access. Panel 3 also supports the unsurprising finding of Figure 4.3 from Chapter 4, that Union Alliances with Occupy movements were more prevalent in cities with larger populations of political liberals.

Figure 5.10 – Rallies, Demonstrations, and Labor Alliances by Size of Liberal Population Interacted with Time

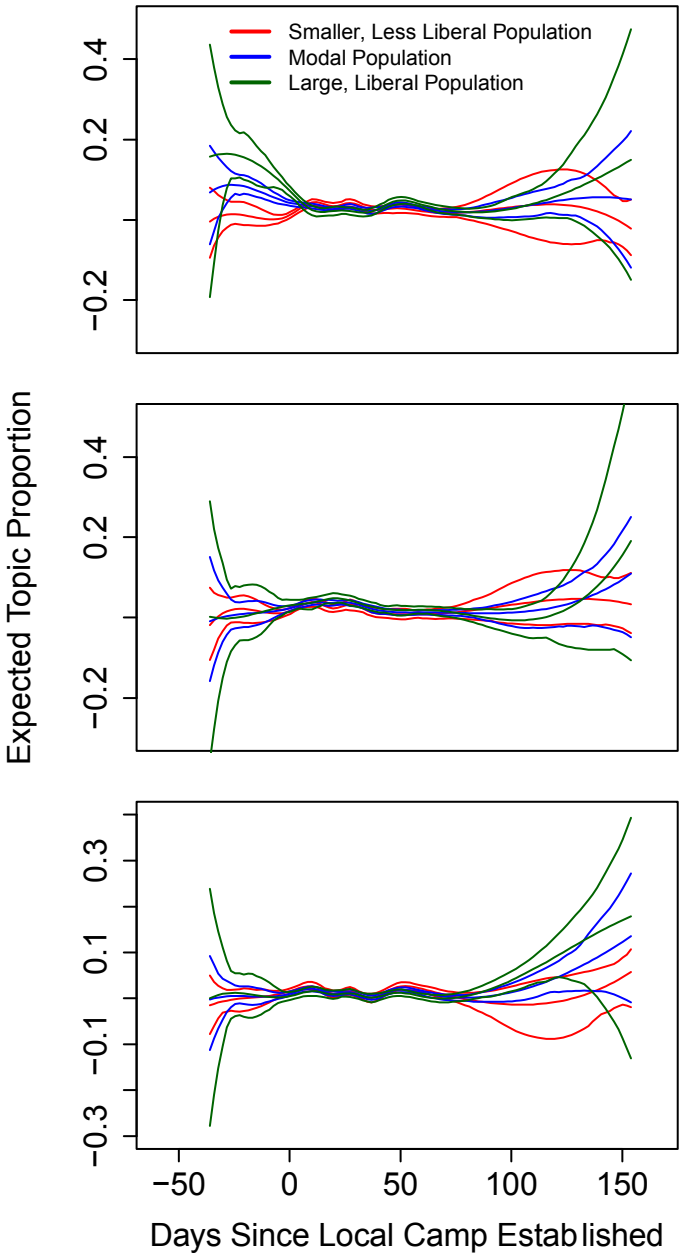


Note: Panel 1 (Top), Panel 2 (Middle), and Panel 3 (Bottom) show the predicted prevalence of Rallies, Demonstrations, and Labor Alliances in the corpus for each day of the Occupy movement. Each panel displays three lines predicting topic prevalence for the highest, lowest, and middle value of the 'number of liberals' variable interacted

with time. See Methodological Appendix for equations used in estimation and prediction.

Figure 5.11, below, shows that protesters in larger cities targeted City Hall sooner than their counterparts in smaller cities (Panel 1), but that campaigns in all cities targeted banks at a similar rate throughout their life courses (Panel 2). Panel 3 shows that these and other actions resulted in sidewalk contestation with police at virtually identical rates and times until late in movements when such battles subsided in smaller cities.

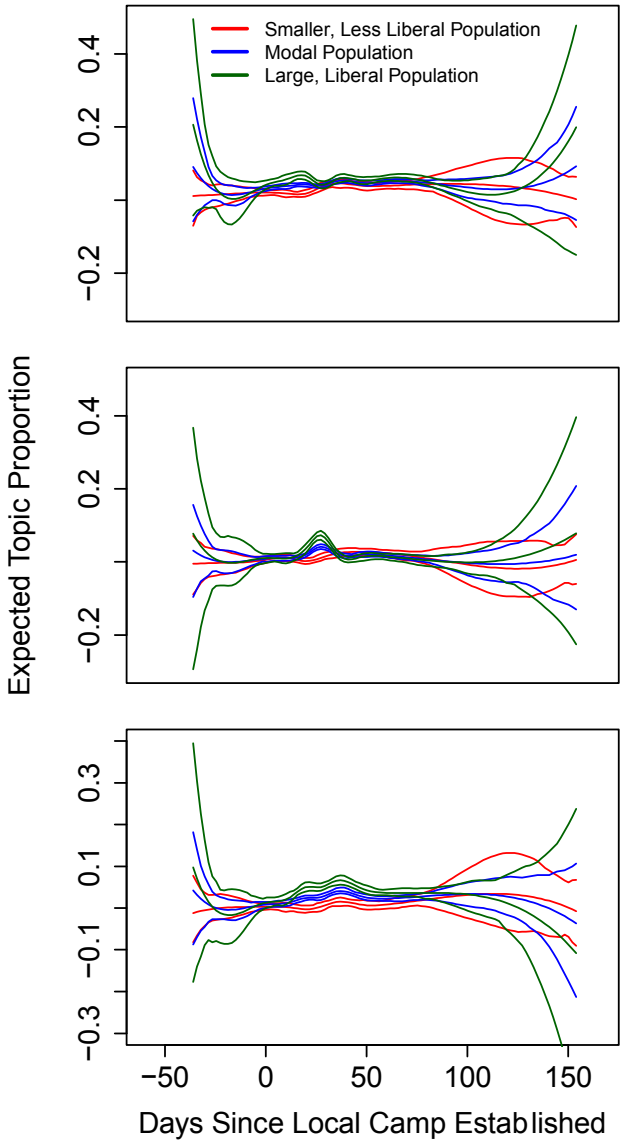
Figure 5.11 – City Hall Targeting, Bank Targeting, and Sidewalk Contestation by Size of Liberal Population Interacted with Time



Note: Panel 1 (Top), Panel 2 (Middle), and Panel 3 (Bottom) show the predicted prevalence of City Hall Targeting, Bank Targeting, and Sidewalk Contestation in the corpus for each day of the Occupy movement. Each panel displays three lines predicting topic prevalence for the highest, lowest, and middle value of the 'number of liberals' variable interacted with time. See Methodological Appendix for equations used in estimation and prediction.

Overall, campaigns in cities with larger pools of support aroused more police response sooner than campaigns in smaller cities. Figure 5.12 Panel 1, below, shows that protesters and police in larger cities engaged in more Traffic Battles sooner and continued the practice at higher levels than their counterparts in smaller cities. Both large and medium cities saw more Curfew Disputes than smaller cities and towns, too. And though virtually all campaigns suffered from arrests in the 7th week of their cause, arrests were more prevalent in medium and larger cities.

Figure 5.12 – Traffic Battles, Curfew Disputes, and Arrests by Size of Liberal Population Interacted with Time



Note: Panel 1 (Top), Panel 2 (Middle), and Panel 3 (Bottom) show the predicted prevalence of Traffic Battles, Curfew Disputes, and Arrests in the corpus for each day of the Occupy movement. Each panel displays three lines predicting topic prevalence for the highest, lowest, and middle value of the 'number of liberals' variable interacted with time. See Methodological Appendix for equations used in estimation and prediction.

Examining topic prevalence as a function of liberal population size through time has revealed some modest differences in Occupy life courses. Cities with larger pools of potential support were quicker to use some performances (like Weekday Marching, Traffic Disruption, and activities Targeting City Hall), and they used them more often. They also experienced more pitched conflict with police. Peaks of some forms of activity came a week or sooner, perhaps, in larger cities. But sequences of activity do not appear to have been entirely re-ordered. So far, this analysis suggests that researchers can credibly speak of a common occupation movement life course with modest, measurable variations.

Centers of Power and Occupy Campaign Life Courses

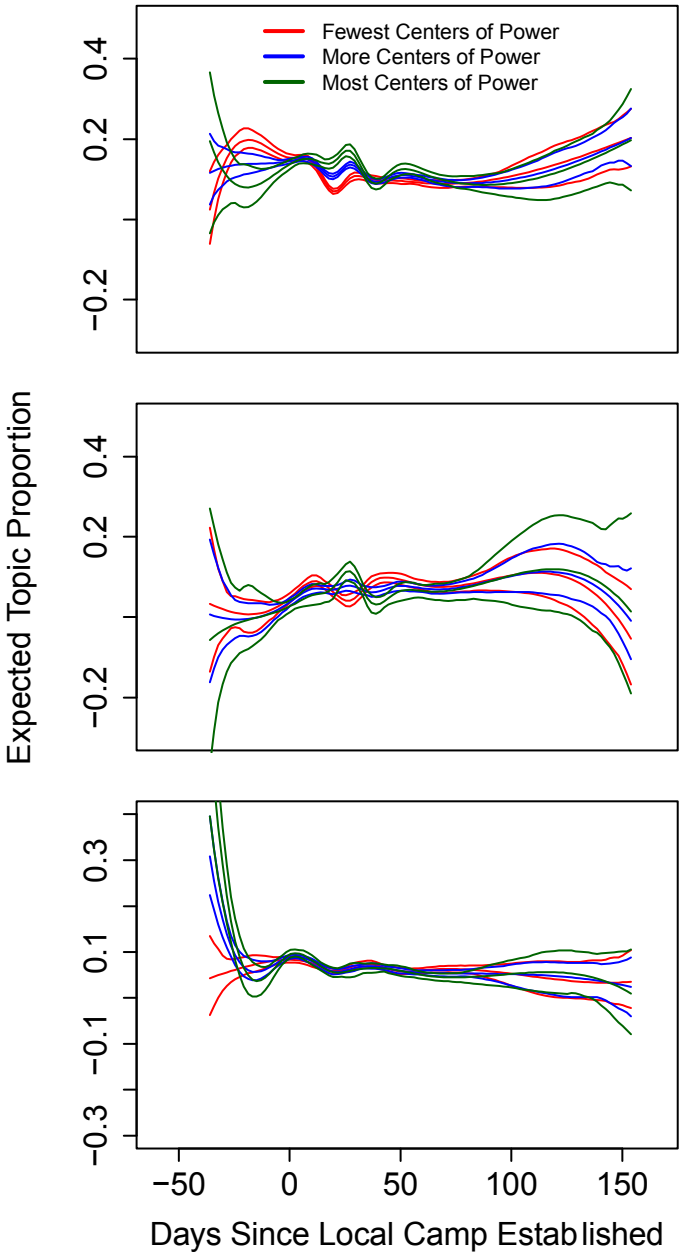
Hypothesis 5.1

The capacity of movements to make change depends not just on the number of citizens they can attract to their cause, but also on their ability to influence decision-making elites who might alter policy to support movements' interests. Mobilizations have been observed to occur more frequently and enjoy more success when political regimes feature more elite power centers with which movements can engage. When there are more power centers in a regime, movement leaders have more potential targets to persuade to their cause, justifying and inspiring the participation of more followers. Any failure to win over a particular member of the political elite, too, is only a partial failure. The movement can still live on, potentially weakening the resolve of a divided government as it persists. If the general theory of political opportunity structures holds in the narrower case of city governments, estimates of performance prevalence as a function of the number of government power centers in a city, through time, should show that movements are more active in cities with more power centers.

Another, understudied linkage between the number of centers of power in a government and the activity levels of protester throughout a campaign concerns protest policing more directly: uni-polar governments may be more decisive, and therefore better able to act strategically and re-take the initiative from protest campaigns. This dimension of Hypothesis 5.1 will be taken up more directly in Chapter 6 as Hypothesis 6.6.

Results displayed in Figures 5.13– 5.16 below show that the number of power centers in a city does correlate with a somewhat different pattern of Occupy activities. While Figures 5.3 and 5.4 only indicated significant difference in the prevalence of Sidewalk Contestion and Curfew Dispute performances, Figure 5.13 show a temporal shift in peak activity for both Weekend Gatherings and Encampment Activities.

Figure 5.13 – Weekend Gatherings, Encampment Activities, Weekday Marches by Number of Power Centers Interacted with Time

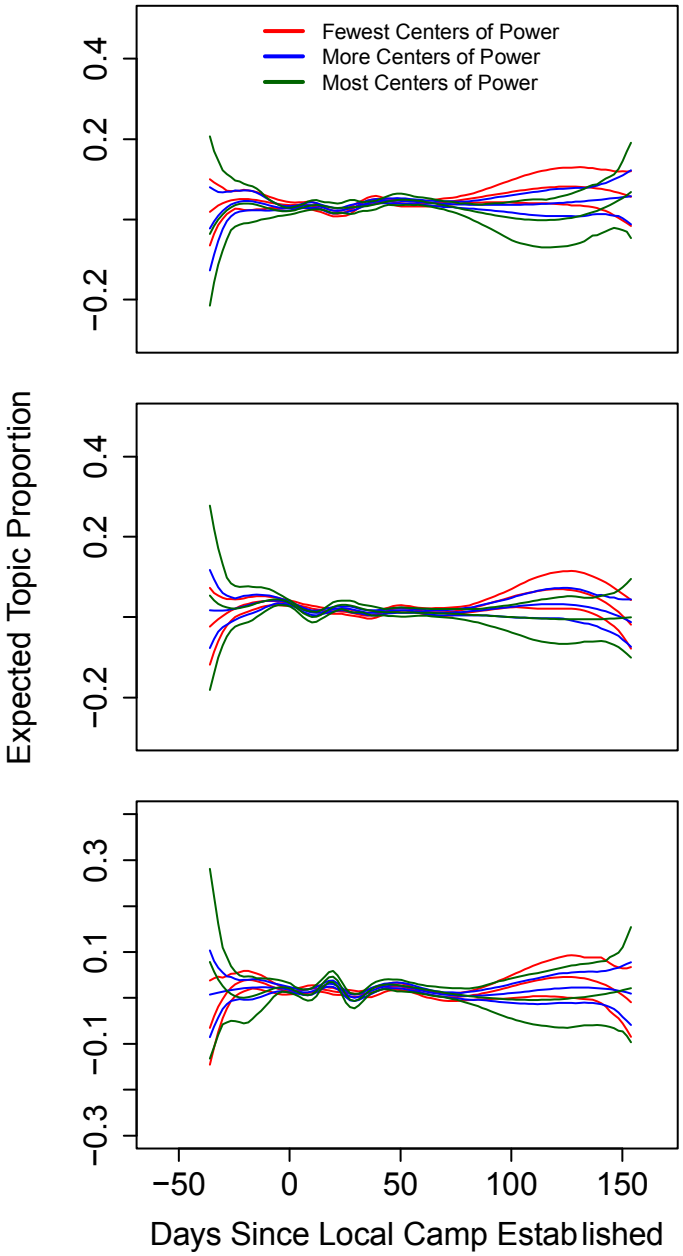


Note: Panel 1 (Top), Panel 2 (Middle), and Panel 3 (Bottom) show the predicted prevalence of Weekend Gatherings, Encampment Activities, Weekday Marches in the corpus for each day of the Occupy movement. Each panel displays three lines predicting topic prevalence for the highest, lowest, and middle value of the 'number of

power centers' variable interacted with time. See Methodological Appendix for equations used in estimation and prediction.

Figures 5.14 and 5.15, apart from the increased prevalence of Sidewalk Contestation in cities with more power centers, show relatively little variation in activities across cities with differing numbers of power centers. But Figure 5.16 shows that Curfew Disputes peaked earlier in cities with power concentrated in a mayor. This finding suggests that differences in Occupy protest activities may owe as much or more to the effect city power concentration has on police activity. Do stronger mayors make for more decisive police? Though this dissertation, in need of a starting point, has treated protesters as prime movers, the dynamic interactions between protesters and police constitute a 'coevolution' over the course of campaigns. Some amount of protester activity is likely caused or repressed by police. Chapters 6 and 7 will take a closer look at the extent to which protesters' *performances* are affected by political opportunity structures' effects on police action. If strong mayors indeed make for decisive police action, it is inappropriate to conclude that differing rates of protester performances are solely the consequence of the structure of static political opportunities.

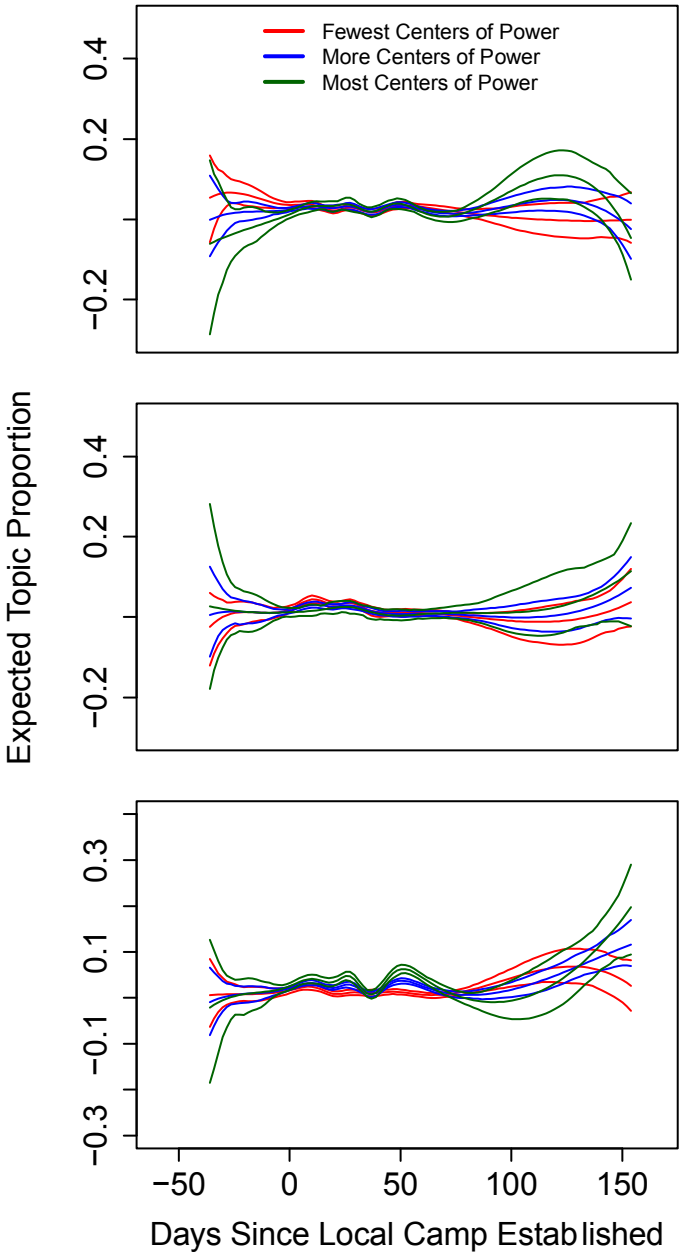
Figure 5.14 – Rallies, Demonstrations, and Labor Alliances by Number of Power Centers Interacted with Time



Note: Panel 1 (Top), Panel 2 (Middle), and Panel 3 (Bottom) show the predicted prevalence of Rallies, Demonstrations, and Labor Alliances in the corpus for each day of the Occupy movement. Each panel displays three lines predicting topic prevalence for the highest, lowest, and middle value of the 'number of power centers' variable

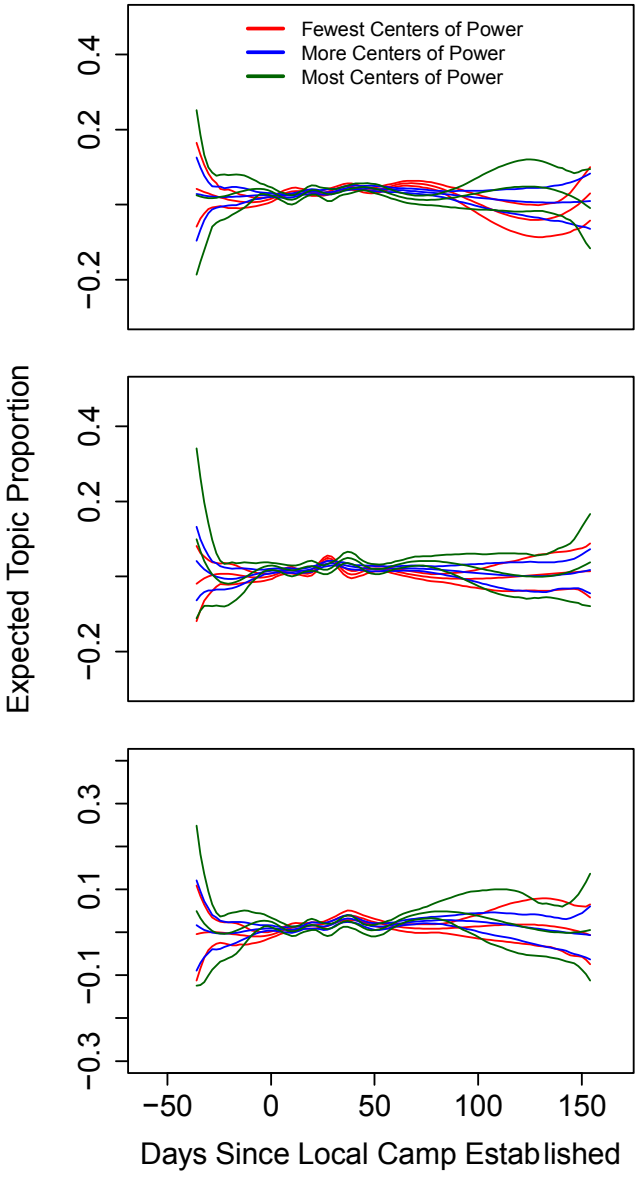
interacted with time. See Methodological Appendix for equations used in estimation and prediction.

Figure 5.15 – City Hall Targeting, Bank Targeting, and Sidewalk Contestation by Number of Power Centers Interacted with Time



Note: Panel 1 (Top), Panel 2 (Middle), and Panel 3 (Bottom) show the predicted prevalence of City Hall Targeting, Bank Targeting, and Sidewalk Contestation in the corpus for each day of the Occupy movement. Each panel displays three lines predicting topic prevalence for the highest, lowest, and middle value of the 'number of power centers' variable interacted with time. See Methodological Appendix for equations used in estimation and prediction.

Figure 5.16 – Traffic Battles, Curfew Disputes, and Arrests by Number of Power Centers Interacted with Time



Note: Panel 1 (Top), Panel 2 (Middle), and Panel 3 (Bottom) show the predicted prevalence of Traffic Battles, Curfew Disputes, and Arrests in the corpus for each day of the Occupy movement. Each panel displays three lines predicting topic prevalence for the highest, lowest, and middle value of the 'number of power centers' variable interacted with time. See Methodological Appendix for equations used in estimation and prediction.

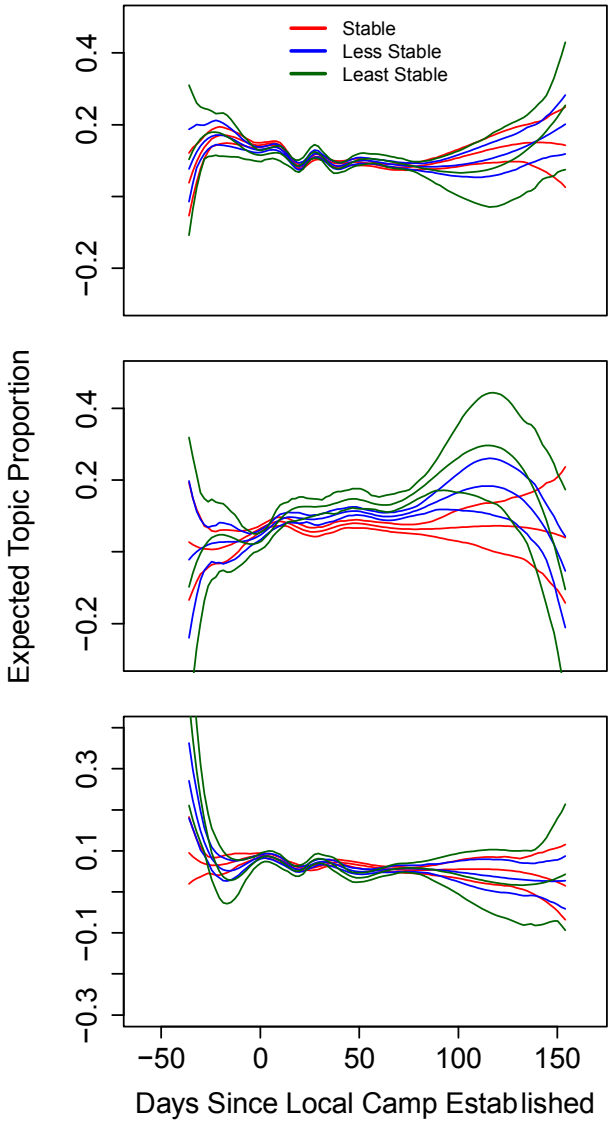
Political Stability and Occupy Campaign Life Courses

Hypotheses 5.2 and 5.3

Whether elected officials are nested in a single chain of command, or sit on multiple cross-cutting committees, all of them have to win votes by engaging their politically active constituents. Since movement leaders and casual marchers know this, researchers should expect that movement campaigns are more active when elections are coming soon. We can also expect that government officials are more accommodating to movements under these conditions, a hypothesis tested here and in Chapter 6. Also, since the unstable political alignments common to newly seated governments generate increased opportunities for movements, we can expect that movements will be more active in cities that just recently underwent an election. These two measures – proximity to prior and upcoming elections have been averaged to create the measure of political stability used in the analyses here. If Tilly's theory holds, less stable governments should experience more overall activity.

Figure 5.17, below, suggests that while city government instability had little impact on Weekend Gatherings (Panel 1), politicians' desires to remain popular with their constituencies may have led them to be more tolerant of Encampment Activities through time (Panel 2).

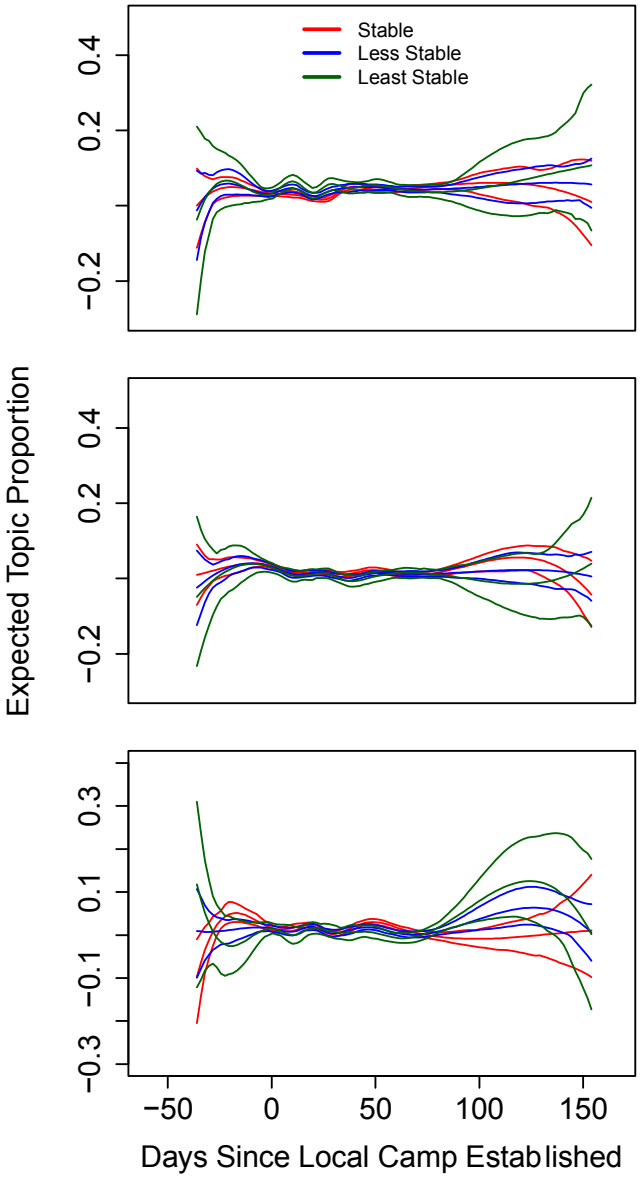
Figure 5.17 – Weekend Gatherings, Encampment Activities, Weekday Marches by Political Instability Interacted with Time



Note: Panel 1 (Top), Panel 2 (Middle), and Panel 3 (Bottom) show the predicted prevalence of Weekend Gatherings, Encampment Activities, Weekday Marches in the corpus for each day of the Occupy movement. Each panel displays three lines predicting topic prevalence for the highest, lowest, and middle value of the 'political instability' variable interacted with time. See Methodological Appendix for equations used in estimation and prediction.

According to Panel 1 of Figure 5.18, below, the least stable governments may have countenanced more Rallies as well. As evidenced by the late rise of the green lines in Panel 3, they may also have made special efforts to fold movement energy into Labor activities near the end of Occupy campaigns.

Figure 5.18 – Rallies, Demonstrations, and Labor Alliances by Political Instability Interacted with Time

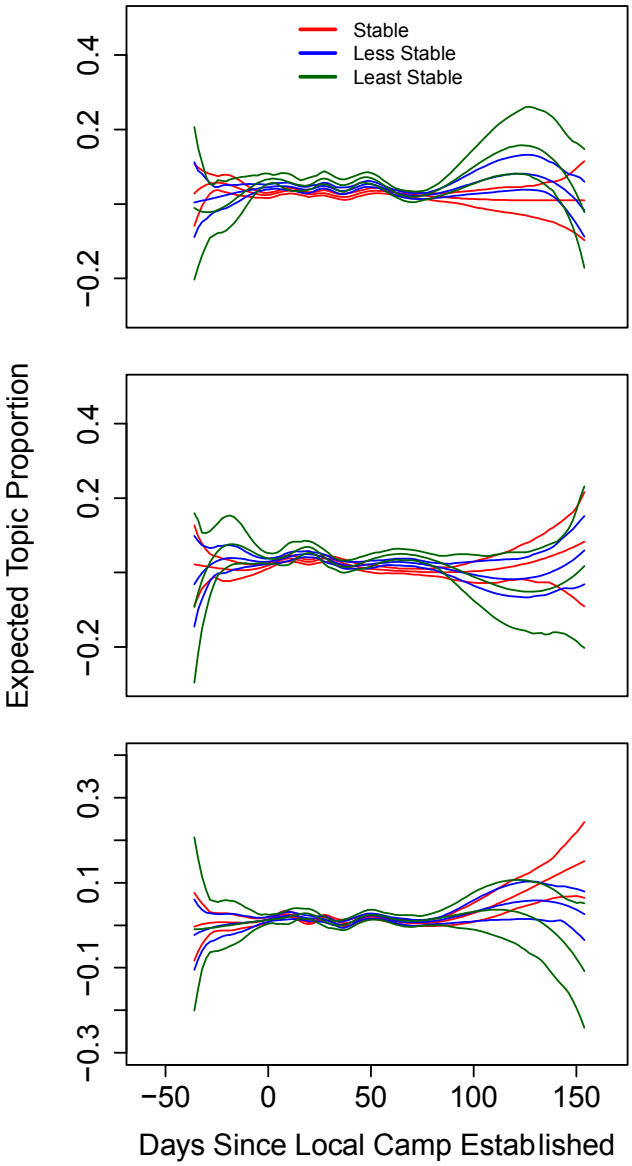


Note: Panel 1 (Top), Panel 2 (Middle), and Panel 3 (Bottom) show the predicted prevalence of Rallies, Demonstrations, and Labor Alliances in the corpus for each day

of the Occupy movement. Each panel displays three lines predicting topic prevalence for the highest, lowest, and middle value of the 'political instability' variable interacted with time. See Methodological Appendix for equations used in estimation and prediction.

As shown in Figure 5.19 Panel 1, Occupy campaigns in the least politically stable cities engaged in significantly more targeting of City Halls. Perhaps protesters in such cities recognized opportunity in that political instability. At least for a moment three weeks into their campaigns, too, they appear to have engaged in more Bank Targeting and Sidewalk Contestation.

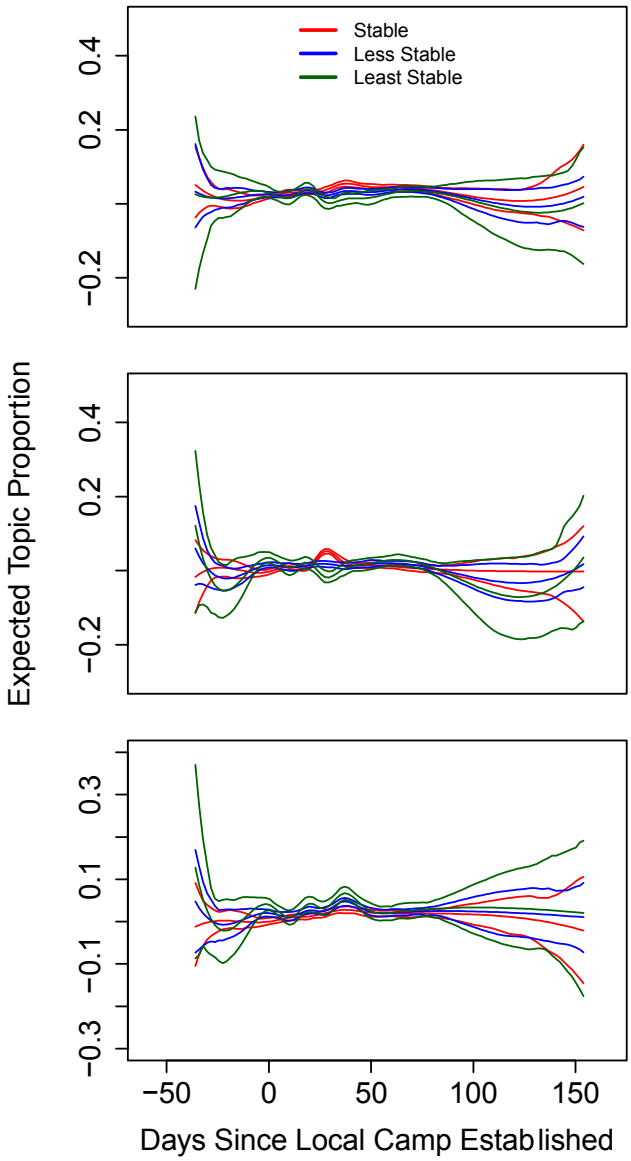
Figure 5.19– City Hall Targeting, Bank Targeting, and Sidewalk Contestation by Political Instability Interacted with Time



Note: Panel 1 (Top), Panel 2 (Middle), and Panel 3 (Bottom) show the predicted prevalence of City Hall Targeting, Bank Targeting, and Sidewalk Contestation in the corpus for each day of the Occupy movement. Each panel displays three lines predicting topic prevalence for the highest, lowest, and middle value of the 'political instability' variable interacted with time. See Methodological Appendix for equations used in estimation and prediction.

City politicians in these municipalities, for their part, seemed to respond in ways that accommodated Occupy campaigns much more than their counterparts in other cities. Politicians up for election or just settling into office may have been less likely to engage protesters in battles over traffic disruption (Panel 1, below). And they appeared significantly less likely to engage in Curfew Disputes at points in campaigns when other cities had decided enough was enough (Panel 2, below).

Figure 5.20 – Traffic Battles, Curfew Disputes, and Arrests by Political Instability Interacted with Time



Note: Panel 1 (Top), Panel 2 (Middle), and Panel 3 (Bottom) show the predicted prevalence of Traffic Battles, Curfew Disputes, and Arrests in the corpus for each day of the Occupy movement. Each panel displays three lines predicting topic prevalence for the highest, lowest, and middle value of the 'political instability' variable interacted with time. See Methodological Appendix for equations used in estimation and prediction.

Conclusions

Some of the hypotheses that motivated this chapter have been confirmed. Political Opportunity Structures did have some impact on Occupy performances.

Campaigns in cities with larger pools of Obama voters were more active as predicted by Hypotheses 5.4. and 5.5.

Political and Electoral Instability affected Occupy campaigns as predicted by Hypothesis 5.2

But the findings here do not support the notion that political opportunity structures significantly altered the sequence or timing of events.[16] Cities with larger or smaller pools of potential supporters may have experienced more or less of some contentious performance, but the basic pattern of activity outlined in Chapter 4 persisted across cities regardless of their support base. Weekend Gatherings kicked off Encampment Activities and Weekday Marches. Rallies and Demonstrations targeting City Halls and Banks persisted at fairly stable levels throughout each campaign. And regardless of city size and political liberalism, Sidewalk Contestation escalated into Curfew Disputes, which escalated into a wave of Arrests. These findings largely confirm Hypotheses 4.5a and 4.5b.

[16] Even where performance prevalence is significantly different across these variables, it is often different at the same moment, easing statistical control.

The exceptions will be discussed further in Chapter 7: Larger cities engaged in the targeting of City Hall a bit sooner than smaller cities. City Halls populated by politicians' whose hold on power was tenuous were targeted more frequently by Occupy campaigns and were less apt to escalate Curfew Disputes or Traffic Battles. But when police in these cities finally did decide to clamp down on their local campaign with arrests, they did it at the same point in the life course as other cities.

These findings are important for two reasons. First, they show that,

> *Occupation campaigns do appear to unfold according to a rather stable life course (with slight variations).*

Researchers have not been able show this in the past. The movements they studied were either too rare or too different to make a life course hypothesis testable.

Second, the key questions of this dissertation (particularly those addressed in Chapter 6) concern the ways in which different police departments policed Occupy movements differently. If the 184 Occupy campaigns of the United States unfolded according to 184, or even 24 different life courses, it would be very difficult to tease out, statistically, the police department characteristics affecting police behavior toward Occupy campaigns. However, since all campaigns appear to have unfolded according to a common life course with only limited variation, the analyses of Chapter 6 will be able to focus on the police department factors that generated disparate protest policing strategies, operations, and tactics. Differences in Occupy campaigns can be statistically accounted for by including 'political instability' and 'number of liberals' control variables in models seeking to explain police behavior.

Chapter 6: Correlates of Protest Policing: Political Context, Police Department Capacity, Police Culture

Chapters 4 and 5 have shown that the 184 Occupy campaigns across the United States engaged in fairly similar sequences of *contentious performances* over their life courses. To the extent that their use of those performances varied, those variations can be accounted for with variables describing the political opportunity structures of the cities in which they were embedded. This chapter, therefore, is able to turn to the questions animating this dissertation: What do police do when they attempt to control protest campaigns? And what factors affect their enactment of *control performances*? Are they behaving in line with the wishes of elites? Are they focused on their own budgets and personnel capacity? Are they enacting the *control performances* that best jive with their department's culture, perhaps accessing available policing heuristics associated with a philosophy of community policing?

Answers to these questions will motivate discussion in Chapter 7 about the extent to which police appear to act out of a sense of *threat to their control*, and how much they act *strategically* or merely *reactively* to protest campaigns.

The Approach and Hypotheses

Chapter 6 pursues a similar analytical path as Chapters 4 and 5. It begins by presenting SVO-amplified topic model results identifying police *control performances*. Then, it regresses city and police department variables on the prevalence of those *control performances* to show how POS, Police Capacity, and Police Culture affected the incidence of *control performances* over the course of an Occupy campaign.

Previous literature on protest policing in the U.S. context has identified several constellations of police activity, often called "approaches," that might well be labeled as *control performances*. Earl, Soule, and McCarthy (2003, 590) created categories for a 'Do Nothing' performance, a 'Nothing to See Here' performance, a 'Legal Eagles' performance, a 'Dirty Harry' performance, and a 'Calling All Cars' performance. The performances, described in Chapter 2, range from showing up and taking no action, to deploying massive force including the use of tear gas, mass arrests, and barricades.

As Chapter 4 did, and as described in Chapter 3, this chapter identifies *performances* not using pre-defined categories, but by allowing a computer algorithm to identify clusters of *actions* (extracted from English grammar) that reliably cohere into *performances.* To test the robustness of this approach, I hypothesize that topic modeling of police *actions* occurring during *police-initiated events* will reveal some common *control performances*. The following hypotheses are based on my own close readings of a thousand or more articles about police and protester activities during the Occupy movements.

Hypothesis 6.1: Topic modeling will recover evidence of control performances warning protesters to cease or modify their activities.

Hypothesis 6.2: Topic modeling will recover evidence of control performances aimed at 'strategic incapacitation,' the citation and ticketing of protesters for violation of minor city ordinances including overnight camping, violating curfews, jaywalking, etc.

Hypothesis 6.3: Topic modeling will recover evidence of control performances that seek to incapacitate and discourage movements by arresting individuals or small groups of protesters (even when police do not target the entire campaign for arrest.)

Hypothesis 6.4: Topic modeling will recover evidence of control performances that seek to peacefully close down encampments.

Hypothesis 6.5: Topic modeling will recover evidence of control performances that seek to close down encampments through large quantities of arrests, police shows of force, and the use of "less lethal" weapons.

Results – Control Performances

Readers will recall from Chapter 4 that topic modeling algorithms generate two forms of results. First, they offer a list of ranked and weighted terms for each topic. Researchers interpret these term lists to determine what a topic is about. The algorithm also yields a measure of the prevalence of each topic in each document. These results allow a researcher to understand what a given document (text unit) is about without reading every last word of it. In this research, results include information about *actions* that constitute topics identifying *control performances*, which, in turn, constitute text units describing *police-initiated events*. Finally, the prevalence of *control performances* across *police-initiated events* can be modeled through time and in light of variables describing the cities and departments in which police work.

As in Chapter 4, this Chapter uses topic modeling to identify *performances* of particular interest for further analysis and hypothesis testing.

Table 6.1 Control Performances Identified by SVO-Amplified LDA

Topic 4 Top Words: ***Ordinance Enforcing***
Highest Probability: city, camping, protesters, police, county, arrested, arrest, others, courthouse, xweekday, ordinance, property, enforce, .m., three
FREX: humboldt, king, subjects, -camping, prohibiting, distribution, ban, county, camping, courthouse, sanchez, ordinances, codes, fps, enforce
Lift: aresheh, carlos, chunk, conceal, craven, kris, now-banned, palmer, police_resist_city_, prohibiting, resisting/obstructing, rivera, rorey, shay, sign-bearing
Topic 7 Top Words: ***Dismantling Camps***
Highest Probability: protesters, police, xpark, tents, xweekday, leave, take, downtown, move, stay, remove, members, arrest, belongings, removed

FREX: downtown, tents_, villa, tents, police_tell_protesters_, members, police_stay_, protesters_leave_, remain, remove, protesters_stay_, space, pack, belongings, warned

Lift: _allow_not/rb_property_, _allow_protesters_remain, _allow_protesters_retrieve, _allow_protesters_stay, _arrest_melissa, _block_xpark_, _bring_city_clean, _bring_the, _bulldoze_an, _cite_anyone, _cite_two_, _close_downtown, _disposition_protesters_, _empty_pancho, _fine_anyone

Topic 8 Top Words: ***Violently Raiding***

Highest Probability: police, protesters, city, polices, gas, riot_gear, unlawful, one, tear, assembly, use, cityname, hall, camp, xpark

FREX: dodger, suits, fired, beanbag, clergy, bullets, gas, affiliate, tear, projectiles, tasha, personnel, stadium, assembly, trap

Lift: _assign_, _pelt_police_, adkison, allusion, antlers, assembly_resist_police_, belts, blame, bull, cain, choya, cops_have_, countless, de-escalate, demobilize

Topic 9 Top Words: ***Deadline Enforcing***

Highest Probability: xpark, p.m., protesters, police, city, time, warning, arrest, deadline, xweekday, eviction, give, night, leave, anyone

FREX: grant, cesar, closure, perry, deadline, notices, closing, notices_, johnathan, p.m., anyone, permit, warning, permits, cease

Lift: _form_protesters_leave, belongings_leave_, bits, bowen_, chilling, clear-, consequences, deadline_come_, deportation, edward, frantic, iconic, immigrant, knx, mountain

Topic 10 Top Words: ***Telling***

Highest Probability: police, tell, said, police_tell_, xweekday, told, sgt., says, say, news, chief, lt., presence, statement, department

FREX: andy, charlie, briefing, cityname-mecklenburg, police_tell_, norwood, solano, vallejo, said, tell, capt., presence, rico, conference, police_tell_police_

Lift: armstrong, bassett, briefing, cityname-mecklenburg, fernandez, neiman, norwood, shirts, _allow_not/rb_gatherings_, _arrest_eight_, _do_monitoring_, _drive_police_, _focus_police_, _form_protesters_, _get_no

Topic 12 Top Words: ***Arresting Resisting Individuals***

Highest Probability: police, one, tents, man, protesters, street, tent, take, arrested, arrest, polices, around, market, person, front

FREX: market, reserve, jessica, kneeling, laser, agitators, third, woman, man, person, chain, basillas, child, adam, man_

Lift: _dismiss_at, _take_he, _use_pepper, advisements, aluminum-frame, avenue_, cannon-like, configuration, description, devin, dirt, disassembling, ellen, fifty-five, full-time

Topic 15 Top Words: ***Arresting Groups***

Highest Probability: people, police, arrested, protesters, xpark, arrest, people_, xweekday, leave, morning, six, two, xweekendday, structure, arrests

FREX: lake, hixon, merritt, shed, people_, six, people, cuesta, structure, d.c., curtis, gentile, mcpherson, main, opposing

Lift: _charge_three, -foot-high, aboard, akard, band, boy, break-, cannon, carrefour, disruption, elderly, exposure, fitzgerald, freeways, fuel

Note: Each Topic is represented by three lists. 'Highest Prob' lists the terms most frequently appearing in the topic. FREX lists terms that are frequent and exclusive to the topic (not used very frequently in other topics). 'Lift' lists terms that are most exclusive to the topic whether or not they are used frequently in the topic. Readers interested in full model output or output alphabetically by topic label may consult Methodological Appendix A.

The analysis of results begins with an interpretation of performance topics by the terms and SVO *action* triplets they comprise.

Topic 10 was the most prevalent in the corpus of *police-initiated events*. It focuses on what "police" "***Tell,"*** "said," or "say" especially to "news" outlets in official "statements" and "briefings."

Topic 7, describing the ***Dismantling of Camps,*** was the next most prevalent *control performance* in the corpus, and involved "police" "taking" "down" and "removing" "protesters'" "tents" in local "parks." Many "protesters" considered "staying", but many were convinced to "leave" rather than risk "arrest" and the forfeiture of their "belongings."

Many times the ***Dismantling of Camps*** was immediately preceded by ***Deadline Warnings and Enforcement***, the subject of **Topic 9.** Topic 9 features "police" "giving" "protesters" "warnings" about an upcoming "eviction" "deadline." They are told to "leave" the "park" by a given "time," on a given "weekday", or face "arrest."

Sometimes, according to **Topic 8,** the *control performances* of Topics 7 and 9 are not enough for police. Enacting the performance of the ***Violent Raid***, "police" show up to "city hall" or a "city park" in "riot gear," declare an "unlawful" "assembly" and "fire" "tear gas"

"beanbags" (presumably rubber) "bullets," and "projectiles" into "trapped" crowds of "protesters."

Whereas the *control performances* identified by Topics 10, 7, 9, and 8 were all directed at Occupy campaigns as a whole. Police also directed many of their *control performances* at individuals or groups of protesters.

Topic 15 – *Arresting Groups* describes groups of "people" being "arrested" by "police." "Police" often performed these "arrests" of "protesters" in "parks" where Occupy campaigns encamped whether it was a "weekday" or a "weekend." Closer inspection of text units in which this topic was prevalent show that groups of protesters from a handful to hundreds were usually arrested after defying police orders to leave encampments.

Police arrests often targeted individual protesters, though. **Topic 12 – *Arresting Resisting Individuals*** describes situations in which "one" "man" is "arrested" by police often for refusing to "take" down "tents" Closer inspection of text units in which this topic was prevalent show that these arrests often included some resistance and scuffle between police and protesters.

Finally, it is clear that police often controlled encampments in the name of city ordinances. Most of these, as described in **Topic 4 – *Ordinance Enforcing***, were prohibitions against "camping." But "police" also "enforced" other "city" and "county" "ordinances" "prohibiting" "sign-bearing," for instance. Deeper inspection of text units associated with topic also show that police cited or "arrested" people for "illegal lodging," issued "criminal trespass notices" to "protesters," and "enforced" "new rules prohibiting the operation of a food distribution table."

Initial Hypotheses Confirmed

As a robustness check of the method of SVO-amplified topic modeling of *control performances*, I ventured 5 hypotheses predicting the *performances* that would be identified by that method.

Hypothesis 6.1 predicted that topic modeling would recover evidence of control performances warning protesters to cease or modify their activities. In fact, this *performance* was the most prevalent in the corpus. Usually through statements and briefings to new media, police signaled, especially early on in campaigns to protesters how they intended to control their *contentious gatherings* and *performances*.[17]

Hypothesis 6.2 predicting that topic modeling would recover evidence of control performances aimed at 'strategic incapacitation,' the citation and ticketing of protesters for violation of minor city ordinances including curfews, jaywalking, etc. was also partially confirmed. Topic 4 focuses on police Ordinance Enforcing.

Hypothesis 6.3 predicting that topic modeling will recover evidence of control performances that seek to incapacitate and discourage movements by arresting individuals or small groups of protesters (even when police do not target the entire campaign for arrest) was likewise confirmed. Topic 12 centers on situations in which police arrest individual protesters, and Topic 15 focuses on situations in which police arrested multiple individuals or groups at a single time. These *control performances* may overlap, in some cases, with those targeting entire Occupy encampments, but this was frequently not the case. Such arrests seemed designed, in many cases, to weaken the campaign without inciting a full-scale battle between police and protesters.

Hypothesis 6.4 predicted that topic modeling would recover evidence of control performances that seek to peacefully close down encampments. This hypothesis was confirmed. Besides 'Telling' protesters what they expected, this was the most prevalent *control performance* of the Occupy movement. As readers will see below, it was used early and often by police.

Hypothesis 6.5 predicted that topic modeling would recover evidence of control performances that seek to close down encampments through large quantities of arrests and the use of "less lethal" weapons. This hypothesis was confirmed. SVO-amplified LDA

[17] See Methodological Appendix A for more details.

(or action-amplified topic modeling, described in full in Chapter 3) recovered a very coherent *control performance* characterized by police wearing riot gear and firing various materiel, from bean bag rounds to tear gas into crowds of protesters as they 'Violently Raided' their encampments.

With a set of inductively identified coherent *control performances* that, while tailored to occupation campaigns, align rather well with the protest policing "approaches" identified in the literature, this chapter moves on to test hypotheses concerning how the prevalence of control performances varies by city and police departmental variables

Prevalence of Control Performances by City and Departmental Variables

This dissertation turns now to its key question: what factors influence police enactment of *control performances*?

As thoroughly discussed in Chapter 2, there are three sets of factors – apart from protesters themselves – that are likely to influence police control performances: the political opportunity structures that shape the interests of political elites who supervise police; the capacity of police departments to act; and the cultures of police departments. As yet, there has been little agreement about how and when these factors influence police *control performances.* Small-N studies are simply not comparable. Different police departments make different choices when facing different social movement campaigns in different political contexts at different times. Large-N studies have been able to show some long-term trends in police use of *control performances.* And they have been able to show that more authoritarian governments use more forceful *control performances* to repress movements compared to their democratic counterparts. But no large-N study has been able to adequately test hypotheses considering how the city government and police department factors affect police allocation of *control performances* in a particular time and place facing a particular movement. None of them have compared across movement *campaigns* to understand how police choices ebb and flow as they interact with a movement.

This chapter uses variables measuring political context, police capacity, and police culture to observe how these factors impact the prevalence of police *control performances* over the life courses of Occupy *campaigns*.

Table 6.2 Independent Variables

Variable Name	**Description**
# centers of power	This variables measures the extent to which political elite power is fractured or concentrated. Across the U.S., three city government types predominate: Mayor-Council, Council-Manager, and Commission. The Mayor-Council type of government features a relatively strong executive and relatively week City Council. In terms of POS, it features the fewest 'independent centers of power." The Council-Manger government type features more powerful council members with their own center of power. And the Commission government type features a number of committees or commissions, each with legislative and executive authority over a specific domain of municipal governance. 'Government type' data have been collected for every city in this dissertation's dataset and arranged on a scale from 1-3, ranging from fewest to most independent centers of power. If a city is also the seat of State power, one point is added to this score. Since no State Capitals are run via Commission, all cities' Government type scores fell between 1 and 3.
Electoral instability	This variable measures the extent to which the local political environment is stable and settled or open to new actors and new coalition formations. I subtracted each Occupy campaign's start date from the date of the next upcoming election. If the next election was within 3 months, the city's 'openness to new actors' score was tallied as a 3; within 3-6 months, a 2, within a year, a 1; beyond a year, 0. Since recent elections also imply that political coalitions are still taking shape. I performed the same procedure to assess the temporal proximity of the camp's start date to the most recent election. I then took a simple average of the two 'instability measures.'

# of liberals	This variable measures the pool of supporters on which Occupy campaigns can draw. It is calculated as City/Town population multiplied by the % of voters casting ballots for Obama in 2008. This continuous variable was then rendered categorical for visualization purposes.
Police budget per capita	This variable measures the available budget of a police department per citizen in its jurisdiction. Collected by the Bureau of Justice Statistics through the Law Enforcement Management and Statistics (LEMAS) survey, this continuous variable was rendered categorical for visualization purposes.
Police per capita	This variable measures the police officers of a police department per citizen in its jurisdiction. Collected by the Bureau of Justice Statistics through the Law Enforcement Management and Statistics (LEMAS) survey, this continuous variable was rendered categorical for visualization purposes.
Community Policing	This variable measures a police department's commitment to Community Policing. The Bureau of Justice Statistics, through the Law Enforcement Management and Statistics (LEMAS) survey, collected over a dozen variables pertaining to community policing. This dissertation uses a simple index of these variables described in the Methodological Appendix – A. This index was then reduced to a 3-value categorical variable for visualization purposes.
Violent Crime	This variable, collected by the Bureau of Justice Statistics, records the violent crime rate for all 184 cities in this dissertation's sample. This continuous variable was rendered categorical for visualization purposes.
% Nonwhite Officers	This variable, collected by the Bureau of Justice Statistics' LEMAS survey, records the % of each department's police force that is ethnically non-white. This continuous variable was rendered categorical for visualization purposes.

Note: All continuous variables – from '# of liberals' through '%nonwhite officers' were divided into nearly equal 'bins' as detailed in the Methodological Appendix A. 'Number of Centers of Power' and 'Electoral Instability' were operationalized according to the specification detailed in Chapter 5.

The variables described in Table 6.2 above are, based on the review of literatures in Chapter 2, are hypothesized to affect police control performances as described in Table 6.3 below.

Recall as well, from Chapter 2, that some of these specific hypotheses may bear on three broader hypotheses particularly relevant to a *police-centered theory* of protest policing. That theory, when seeking explanations for protest policing (including police enactment of control performances), takes as its point of departure the perspective of police officers and departments. From that perspective, decisions about protest policing are often made based on police evaluations of threats to their control over situations - a driving concern of anyone whose daily interactions involve significant risk, uncertainty, and the possibility that they might effectively manage the two.

The police-centered theory also generates two other hypotheses. One, advanced by Earl, Soule, and Davenport, suggests that U.S. police are largely insulated from political elites and their concerns, and therefore choose *control performances* almost entirely in the moment, in reaction to protesters *contentious performances.* This dissertation advances an alternative hypothesis, that police are strategic, choosing *control performances* based not only on protester threats, but also the likelihood that their choices will sit well with supervising political elites. Moreover, strategic police choose the timing of their *control performances* to maximize their strategic objectives; sometimes they even take the initiative to stymie protesters and maintain control. Column 4 of Table 6.3 notes whether or not a hypothesis is relevant to a broader Police-Centered theory.

Table 6.3 Hypotheses: Effects of City and Police Variables on Control Performances

Hypothesis	Description	Variables	Police-Centered Hypothesis?
6.6	Police performances will be more ambivalent to campaigns in cities where elites are fractured across many power centers.	# centers of power	Strategic, responding to elites
6.7	Police will be more accommodating to movements where electoral instability makes elites	Electoral instability	Strategic, responding to

	accommodating to voters.		elites
6.8	Police will use more Individual arrests to keep their *control performances* out of the headlines.	Electoral instability	Strategic, responding to elites
6.9	Police performances will reflect their beliefs about the popularity of the movement with elected officials' constituencies. Departments in smaller more conservative towns will be less accommodating to Occupy campaigns.	# of liberals	Strategic, responding to elites
6.10	Police performances will reflect a sense of threat from larger crowds in liberal cities	# of liberals	Threatened
6.11	Police with smaller budgets are more likely to see long-lasting campaigns as a threat to their budgets, and may be more likely to shut down those campaigns (by any means necessary) sooner.	Police budget per capita	Threatened
6.12	Like departments with smaller budgets, departments with fewer officers per capita are more likely to see long-lasting campaigns as a threat to their workforce capacity, and may be more likely to shut down those campaigns (by any means necessary) sooner.	Police per capita	Threatened
6.13	Since these departments may have a general sense of being under-powered, they may prefer Arresting Individuals as opposed to performing group Arrests.	Police per capita	Threatened

6.14	Police in departments committed to community policing will be more accommodating to movements in general.	Community Policing	
6.15	Since a tenant of community policing involves avoiding group punishments for individual actions, departments with this philosophy are likely to apply control to individuals more often than to camps.	Community Policing	
6.16	Departments in cities with high violent crime rates are less likely to view protesters as a major threat or priority, and therefore are likely to respond slower and with more accommodation.	Violent Crime	Threatened
6.17	Departments with more non-white officers are generally less-likely to take a hardline against social justice movements, and so will be more accommodating to Occupy campaigns.	Nonwhite Officers	

Results from Structural Topic Models

Structural Topic Models regress independent variables across topic prevalence scores generated by LDA topic modeling. The models used in this chapter to investigate hypotheses about the prevalence *and* timing of control performances use the following general equation:

$$\gamma = \tau(x) + \beta_1 \times \tau(x) + \beta_2(x) + \beta_3(x) + \cdots \beta_n(x) + \varepsilon$$

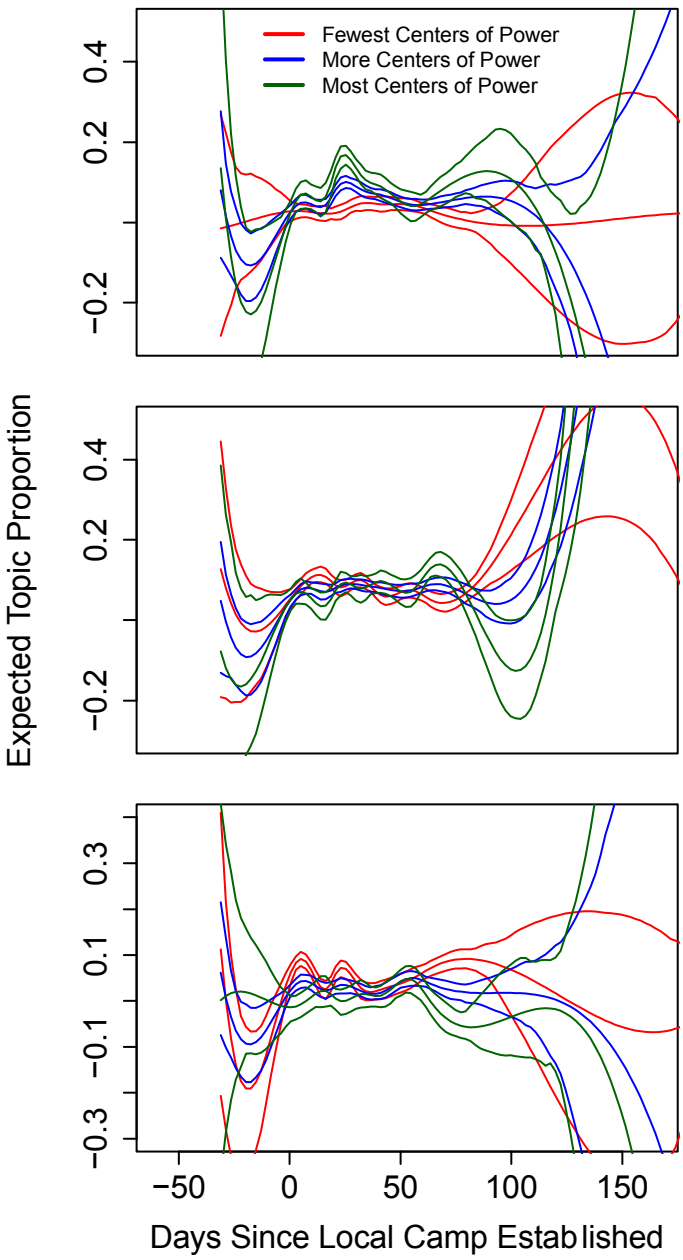
Readers may recall this general equation from Chapter 5. As in the latter models and figures of Chapter 5 the above equation is used to estimate $\beta_1, \beta_2, \beta_3, \ldots \beta_n$ then predict values of γ across three different values of one β for τ while holding the other β coefficients (e.g. β_2, β_3) at their means.

Unlike the models and figures of Chapter 5, however, Figures 6.1-6.14 use a larger array of control variables. Since both city-level POS variables *and* variables describing police departments are likely to affect police-initiated control performances, all these variables are used in each model. Again, as in Figures 4.8 – 5.20, all but one of these variables are held at their means when predicting γ for display in figures. And as in Figures 5.9 – 5.20, one β of interest is interacted with τ, then predictions of γ are displayed for each value of the β of interest across τ.

In Figures 6.1 – 6.12 below, we see how city and police department variables affected the prevalence of *control performances* for each day of Occupy encampments.

Hypothesis 6.6 predicts that police will act more decisively sooner in cities with fewer centers of political power. If decisive action is measured as the use of forceful control performances like Arresting, Figure 6.1 seems to support that hypothesis. Cities with relatively more power vested in a mayor were much more like to forego the softer approach of Ordinance Enforcing, and launch directly into Group and Individual Arresting earlier in Occupy campaigns. Cities with more power centers only picked up their arrest rates near the end of Occupy campaigns, relying instead on a strategy of hassling individual Occupiers with fines, citations, and eviction notices.

Figure 6.1 – Ordinance Enforcing, Group Arresting, Arresting Resisting Individuals by Number of Power Centers Interacted with Time

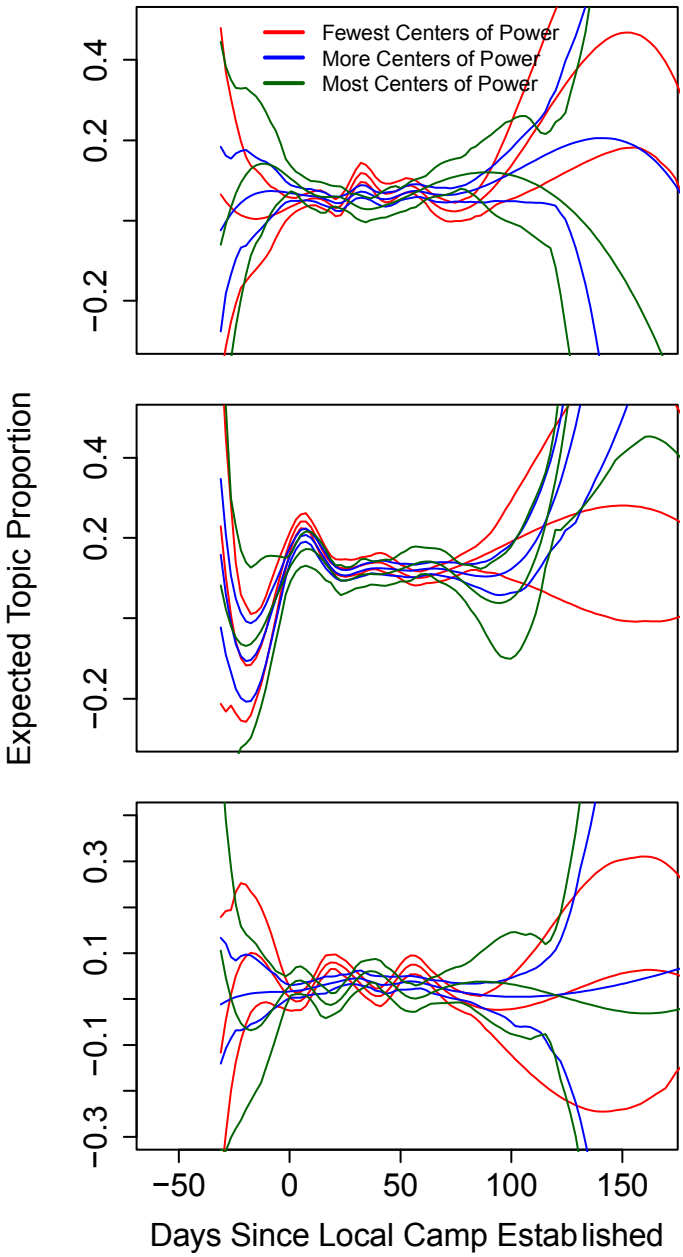


Note: Panel 1 (Top), Panel 2 (Middle), and Panel 3 (Bottom) show the predicted prevalence of Ordinance Enforcing, Group Arresting, Arresting Resisting Individuals in the corpus of text units describing police-initiated events for each day of the Occupy movement. Each panel displays three lines predicting control performance prevalence for the highest, lowest, and middle value of the 'number of power centers' variable interacted with time. See Methodological Appendix for equations used in estimation and prediction.

Cities with Fewer power centers, too, were more likely to shut down encampments immediately upon their establishment as shown in Panel 2 of Figure 6.2, below. They

performed more Raids, more frequently than police departments in other cities, as well, as shown in Panel 3, below.

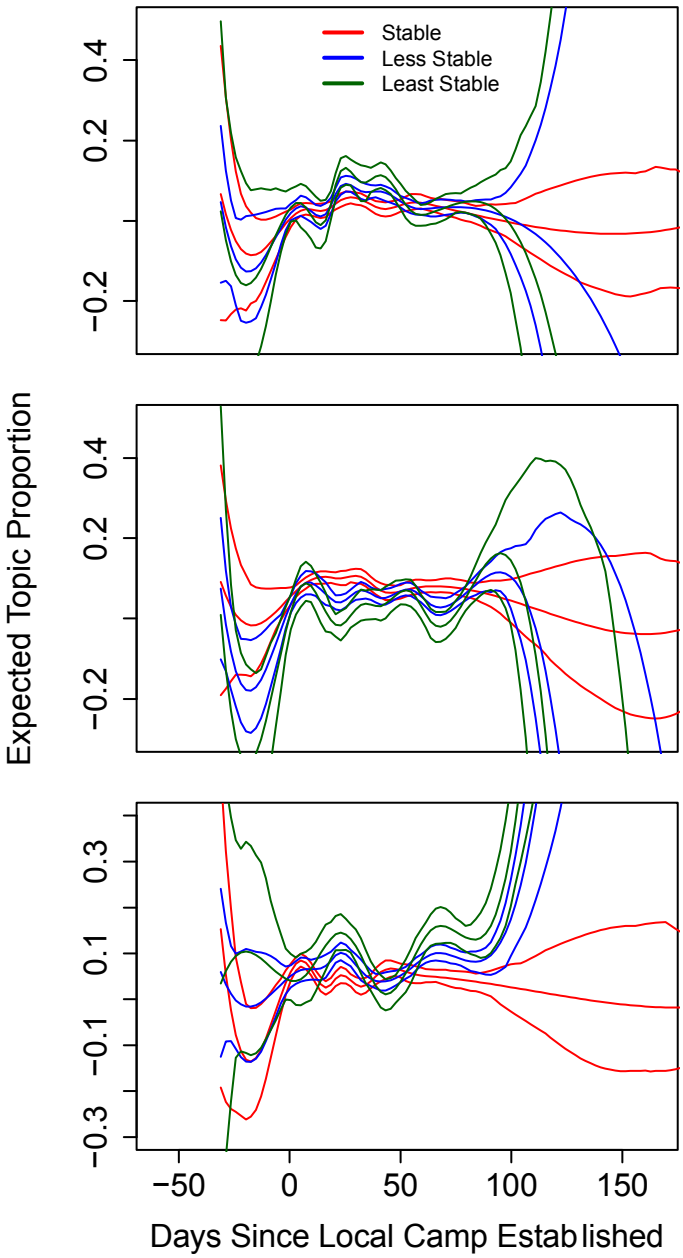
Figure 6.2 – Deadline Enforcing, Dismantling Camps, and Violent Raiding by Number of Power Centers Interacted with Time



Note: Panel 1 (Top), Panel 2 (Middle), and Panel 3 (Bottom) show the predicted prevalence of Deadline Enforcing, Dismantling Camps, and Violent Raiding in the corpus of text units

describing police-initiated events for each day of the Occupy movement. Each panel displays three lines predicting control performance prevalence for the highest, lowest, and middle value of the 'number of power centers' variable interacted with time. See Methodological Appendix for equations used in estimation and prediction.

Figure 6.3 – Ordinance Enforcing, Group Arresting, Arresting Resisting Individuals by Political Instability Interacted with Time

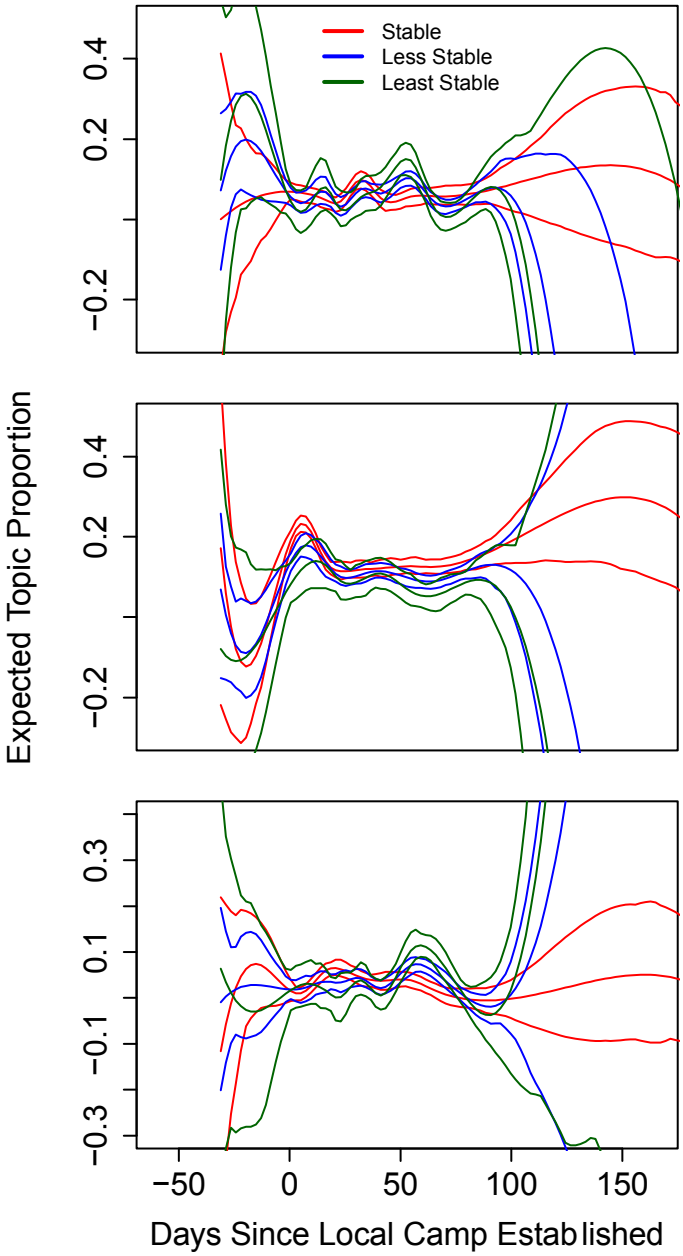


Note: Panel 1 (Top), Panel 2 (Middle), and Panel 3 (Bottom) show the predicted prevalence of Ordinance Enforcing, Group Arresting, Arresting Resisting Individuals in the corpus of text units describing police-initiated events for each day of the Occupy movement. Each panel displays three lines predicting control performance prevalence for the highest, lowest, and middle value of the 'instability' variable interacted with time. See Methodological Appendix for equations used in estimation and prediction.

Figure 6.3, above, shows that police in the least politically stable cities were somewhat more likely to use tactics of strategic incapacitation, including hassling protesters with the energetic enforcement of city ordinances. They were less likely to make group arrests than their counterpart forces in cities that were more politically stable. But when they did make arrests, they were more likely to target individuals and result in scuffles.

If police in politically unstable cities sought to discourage movements with citations, their counterparts appear to have been more willing to reject camping at the outset of Occupy campaigns. Panel 2 of Figure 6.4, below, shows that departments in stable cities were significantly more likely to simply remove tents as soon as they were pitched. Panel 3 shows that they raided camps sooner, too. Departments in less stable cities, on the other hand, attempted to set and enforce deadlines multiple times and then engaged in Violent Raiding of camps at a significantly higher rate later in the course of campaigns.

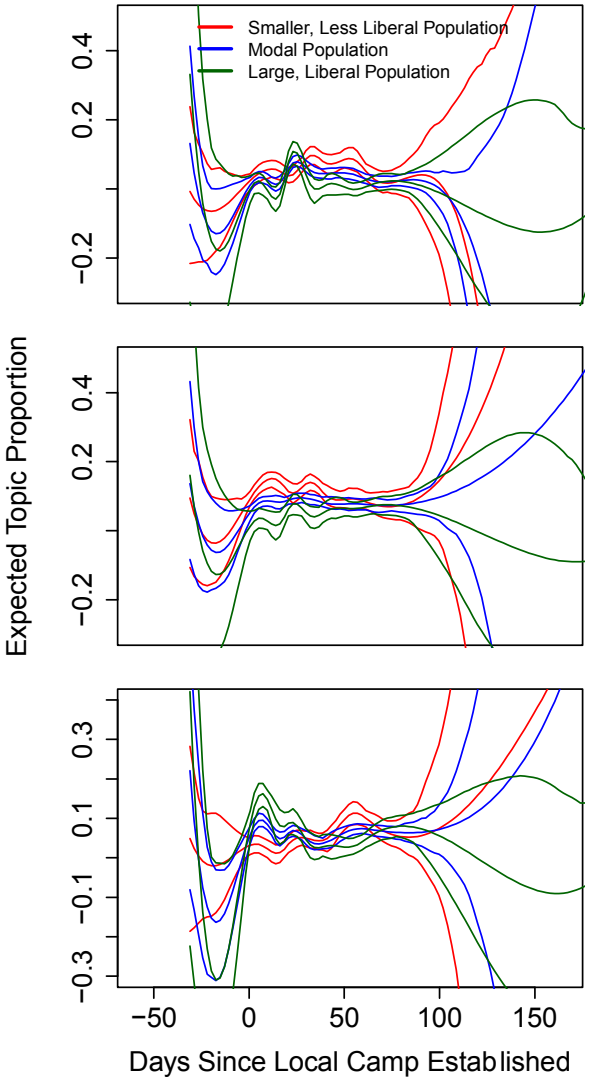
Figure 6.4 – Deadline Enforcing, Dismantling Camps, and Violent Raiding by Political Instability Interacted with Time



Note: Panel 1 (Top), Panel 2 (Middle), and Panel 3 (Bottom) show the predicted prevalence of Deadline Enforcing, Dismantling Camps, and Violent Raiding in the corpus of text units describing police-initiated events for each day of the Occupy movement. Each panel displays three lines predicting control performance prevalence for the highest, lowest, and middle value of the 'political instability' variable interacted with time. See Methodological Appendix for equations used in estimation and prediction.

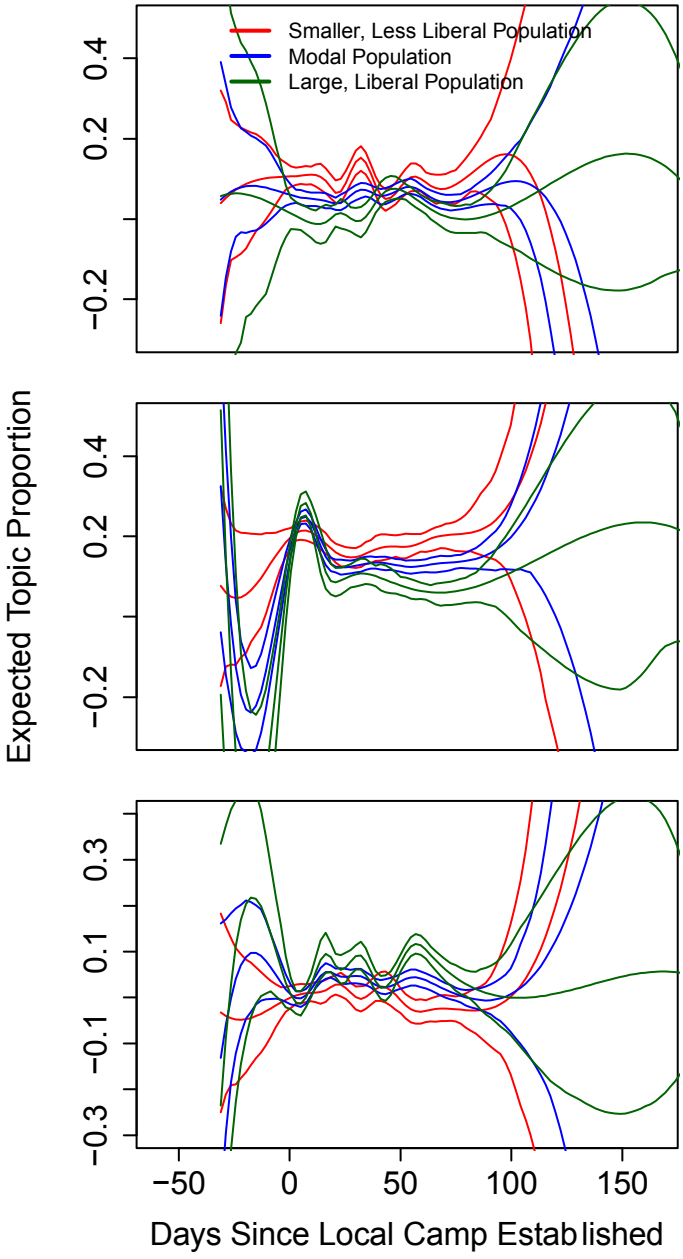
Figure 6.5, below shows that police in jurisdictions with fewer liberals were certainly unwelcoming to Occupy campaigns throughout their life course. They Enforced more Ordinances, and engaged in more Group Arrests throughout the fall of 2011. They also, according to Figure 6.6 set and enforced Deadlines early and often and had no qualms about persistently dismantling camps. This steady, unwelcoming pressure seems to have discouraged campers so much that these cities did not need to engage in the sort of Raids that closed down most camps in cities that may have initially encouraged or accommodated the movement.

Figure 6.5 – Ordinance Enforcing, Group Arresting, Arresting Resisting Individuals by Size of Liberal Population Interacted with Time



Note: Panel 1 (Top), Panel 2 (Middle), and Panel 3 (Bottom) show the predicted prevalence of Ordinance Enforcing, Group Arresting, Arresting Resisting Individuals in the corpus of text units describing police-initiated events for each day of the Occupy movement. Each panel displays three lines predicting control performance prevalence for the highest, lowest, and middle value of the 'number of liberals' variable interacted with time. See Methodological Appendix for equations used in estimation and prediction.

Figure 6.6 – Deadline Enforcing, Dismantling Camps, and Violent Raiding by Size of Liberal Population Interacted with Time



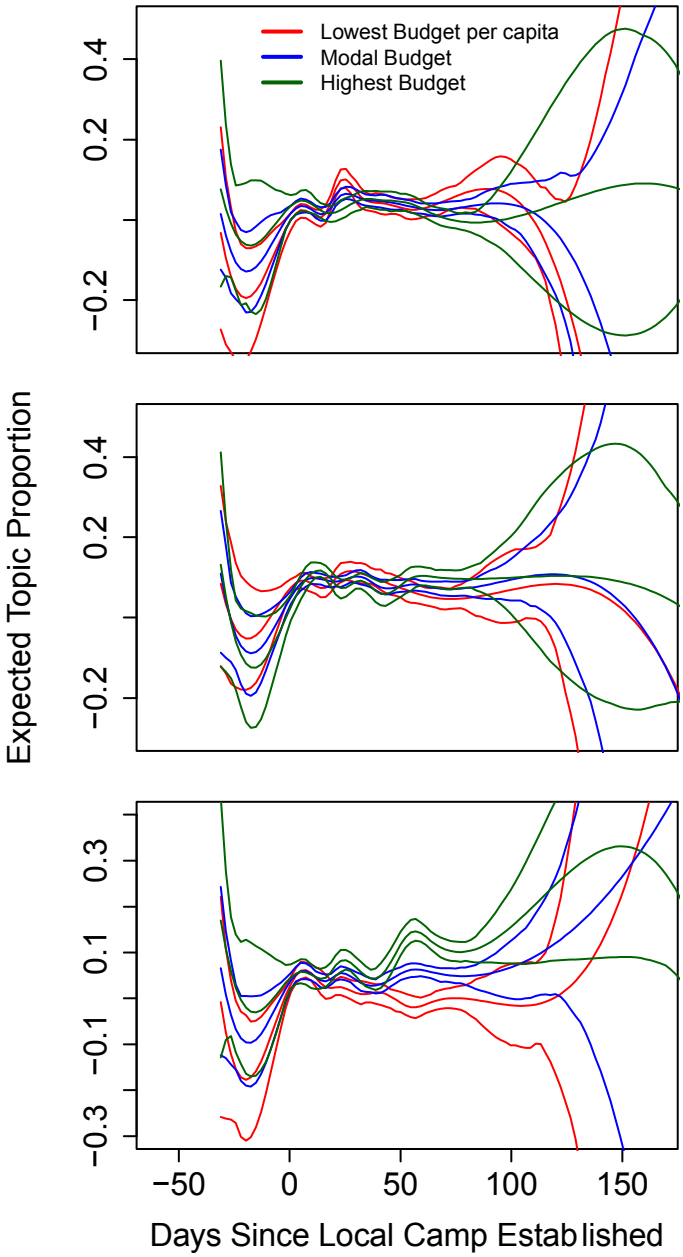
Note: Panel 1 (Top), Panel 2 (Middle), and Panel 3 (Bottom) show the predicted prevalence of Ordinance Enforcing, Group Arresting, Arresting Resisting Individuals in the corpus of text units describing police-initiated events for each day of the Occupy movement. Each panel displays three lines predicting control performance prevalence for the highest, lowest, and middle value of the 'number of liberals' variable interacted with time. See Methodological Appendix for equations used in estimation and prediction.

Police Capacity

Earl and her coauthors suggest that U.S. departments are rather autonomous from Cities, at least compared to departments in many other developed countries. If this is true, police capacities and cultures should have significantly more impact on police control performances than city-level variables we have just reviewed.

The analysis of police department variables' effects on their policing of protest begins with an evaluation of the effect of police budgets on department's control performances. Figures 6.7 and 6.8 show that department budget per capita had a modest impact on control performances. Poorer departments appear to have been somewhat more likely to Enforce Ordinances against protesters (Figure 6.7, Panel 1), perhaps as a revenue-generating tactic. While budget seems to have had only a small impact on the use of Arrests, with better-off departments, perhaps engaging in that control performance sooner, budget had a significant impact on the prevalence of Individual Arrests. It appears that department with greater access to resources were much, much more likely to single out Individuals. Was this a strategy used to incapacitate protest campaigns without arousing major force-on-force battles? And is it a strategy that is especially expensive to carry out? This possibility will be discussed further in Chapter 7.

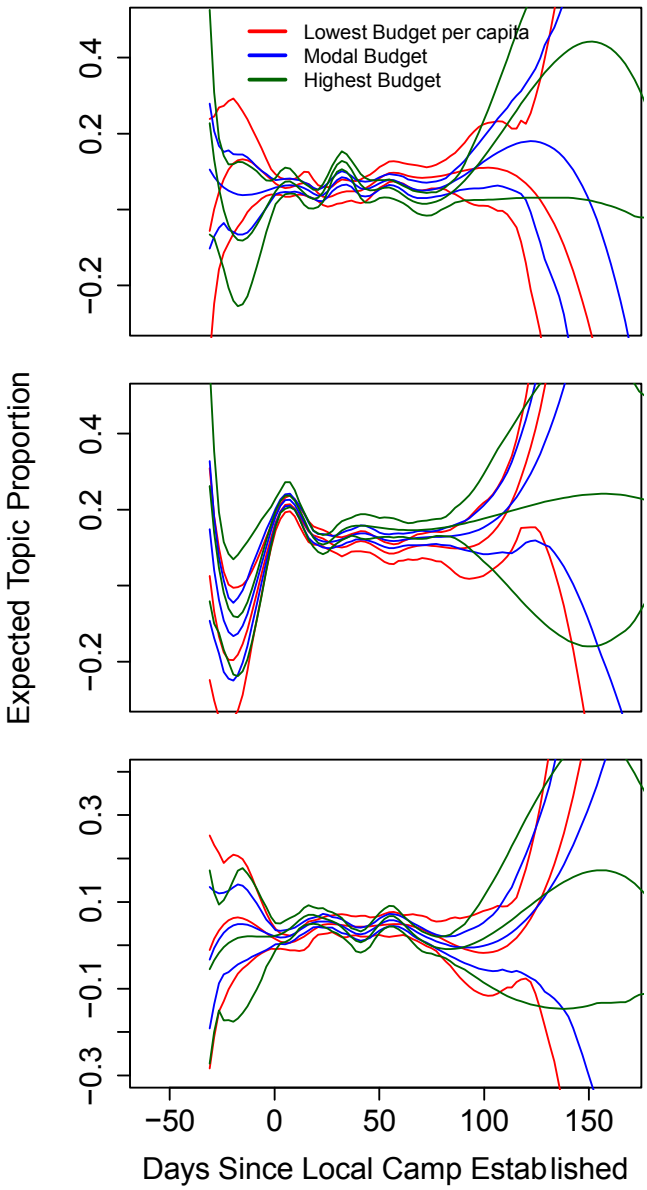
Figure 6.7 – Ordinance Enforcing, Group Arresting, Arresting Resisting Individuals by Police Budget Interacted with Time



Note: Panel 1 (Top), Panel 2 (Middle), and Panel 3 (Bottom) show the predicted prevalence of Ordinance Enforcing, Group Arresting, Arresting Resisting Individuals in the corpus of text units describing police-initiated events for each day of the Occupy movement. Each panel displays three lines predicting control performance prevalence for the highest, lowest, and middle value of the 'police budget' variable interacted with time. See Methodological Appendix for equations used in estimation and prediction.

If budgets affect the way departments handle individual protesters, they seem to have relatively little effect on how they manage protest campaigns. Better-resourced departments (as shown in Panel 1 of Figure 6.8) may have been better able to establish and enforce deadlines on camps. They also appear to have taken direct action to Dismantle Camps more frequently than relatively poor departments (Panel 2).

Figure 6.8 – Deadline Enforcing, Dismantling Camps, and Violent Raiding by Police Budget Interacted with Time

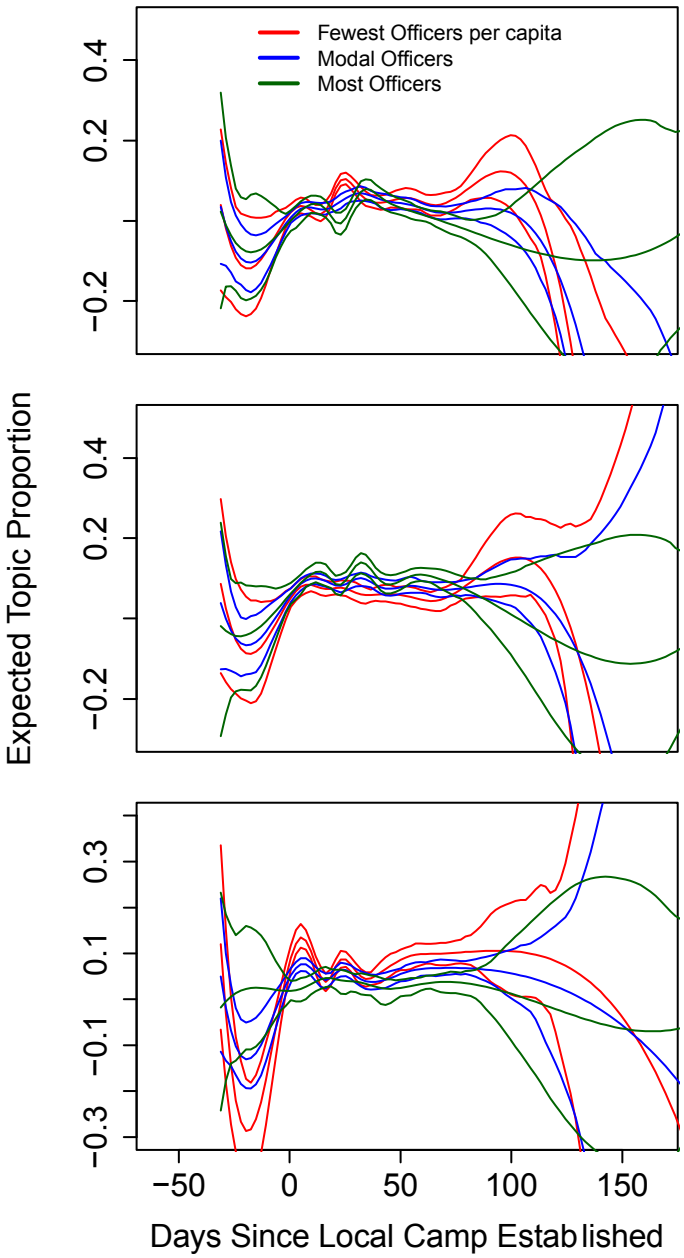


Note: Panel 1 (Top), Panel 2 (Middle), and Panel 3 (Bottom) show the predicted prevalence of Ordinance Enforcing, Group Arresting, Arresting Resisting Individuals in the corpus of text units describing police-initiated events for each day of the Occupy movement. Each panel displays three lines predicting control performance prevalence for the highest, lowest, and middle value of the 'police budget' variable interacted with time. See Methodological Appendix for equations used in estimation and prediction.

Figure 6.9, below, shows that departments with relatively few officers per capita were more likely to issue citations earlier during movements (Panel 1), perhaps to establish

their control over those camps. When they engaged in arrests, they were more likely to target individuals, too (Panel 3).

Figure 6.9 – Ordinance Enforcing, Group Arresting, Arresting Resisting Individuals by Officers per Capita Interacted with Time

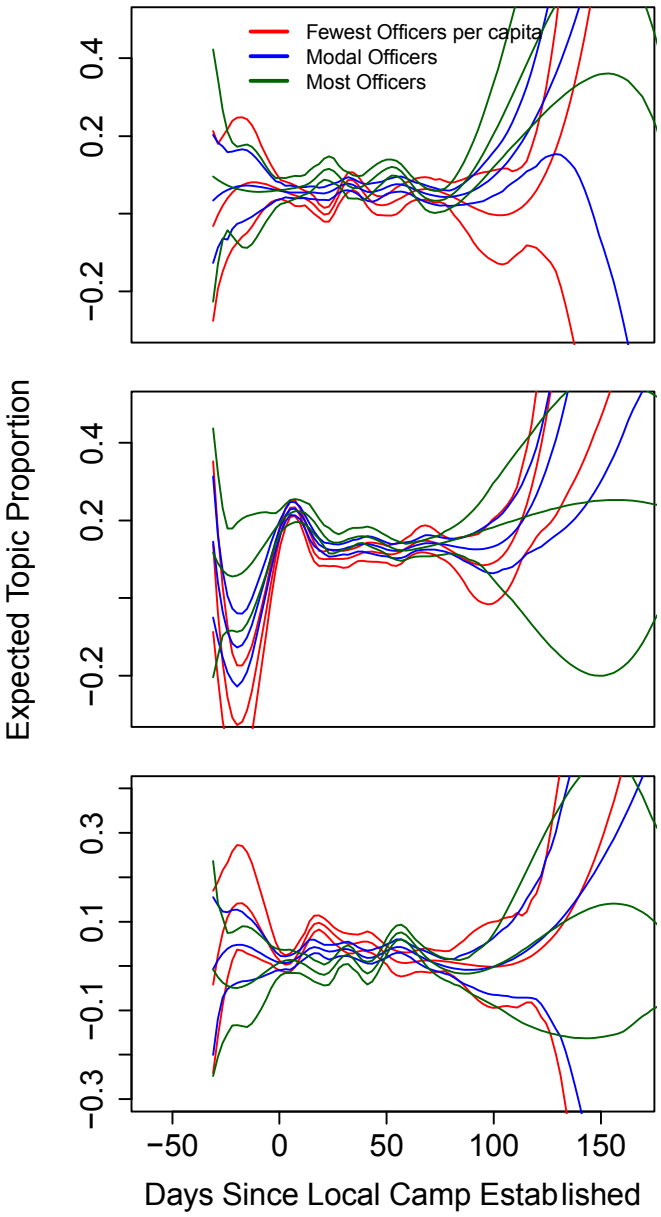


Note: Panel 1 (Top), Panel 2 (Middle), and Panel 3 (Bottom) show the predicted prevalence of Ordinance Enforcing, Group Arresting, Arresting Resisting Individuals in the corpus of text units describing police-initiated events for each day of the Occupy movement. Each panel displays three lines predicting control performance prevalence for the highest, lowest, and

middle value of the 'officers per capita' variable interacted with time. See Methodological Appendix for equations used in estimation and prediction.

Departments with relatively few officers may have an ongoing sense that they have to do a lot with a little, that they need to be strategic about their operations. Some indication of that attitude appears to hold in the case of protest policing. While larger forces were establishing Deadlines for camps (Panel 1), smaller forces (relative to their citizen population) were raiding camps. Larger departments were more tolerant of camps, seeking to remove them with deadlines before carrying out a wave of raids in week 8 of encampments.

Figure 6.10 – Deadline Enforcing, Dismantling Camps, and Violent Raiding by Officers per Capita Interacted with Time



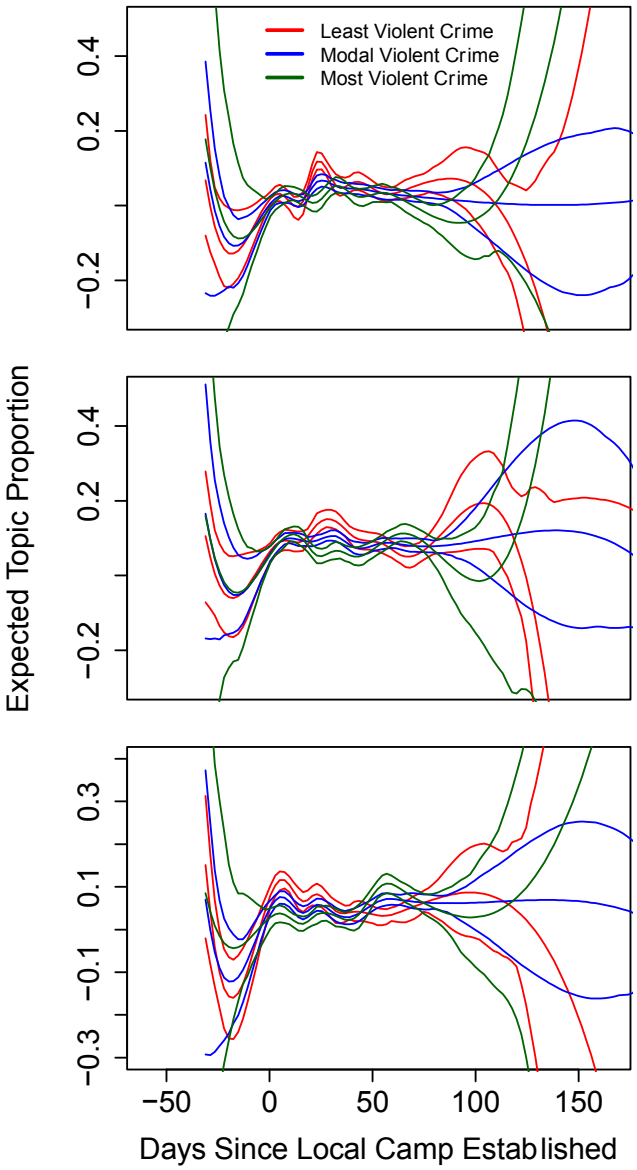
Note: Panel 1 (Top), Panel 2 (Middle), and Panel 3 (Bottom) show the predicted prevalence of Ordinance Enforcing, Group Arresting, Arresting Resisting Individuals in the corpus of text units describing police-initiated events for each day of the Occupy movement. Each panel displays three lines predicting control performance prevalence for the highest, lowest, and middle value of the 'officers per capita' variable interacted with time. See Methodological Appendix for equations used in estimation and prediction.

Police Culture

Police department's capacities appear to affect department's strategic calculus as shown in Figure 6.10. Other aspects of a department are likely to affect its views of protest as well.

Figure 6.11 shows that departments in cities with high violent crime rates may have had a relatively blasé attitude toward protest policing. Such departments were significantly less likely to engage in Ordinance Enforcing. They also arrested fewer protesters until later in their campaigns.

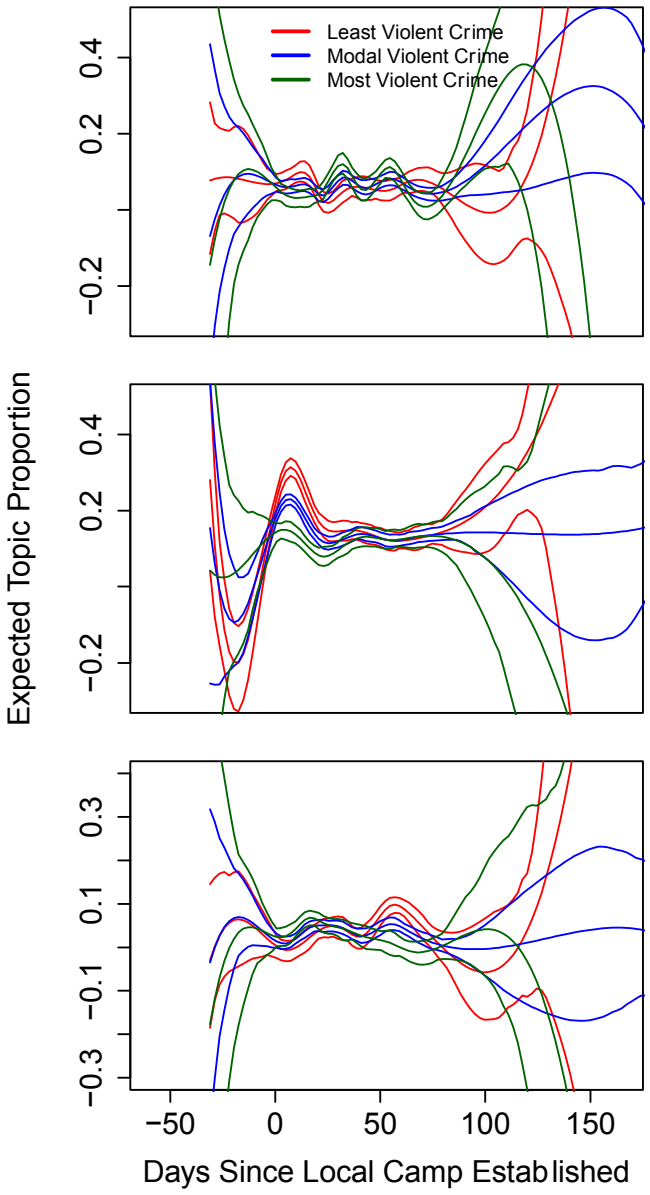
Figure 6.11 – Ordinance Enforcing, Group Arresting, Arresting Resisting Individuals by Police Experience with Violent Crime Interacted with Time



Note: Panel 1 (Top), Panel 2 (Middle), and Panel 3 (Bottom) show the predicted prevalence of Ordinance Enforcing, Group Arresting, Arresting Resisting Individuals in the corpus of text units describing police-initiated events for each day of the Occupy movement. Each panel displays three lines predicting control performance prevalence for the highest, lowest, and middle value of the 'police experience with violent crime' variable interacted with time. See Methodological Appendix for equations used in estimation and prediction.

Departments where homicides and assaults were more common were also slower to react to Occupy campaigns. They were much less likely to immediately Dismantle encampments (Panel 2 of Figure 6.12), tending instead to manage campaigns through Deadlines threats (Panel 1). Eventually, these departments relied on arrests to close down their camps (Panels 2 & 3 of Figure 6.11, above).

Figure 6.12 – Deadline Enforcing, Dismantling Camps, and Violent Raiding by Police Experience with Violent Crime Interacted with Time

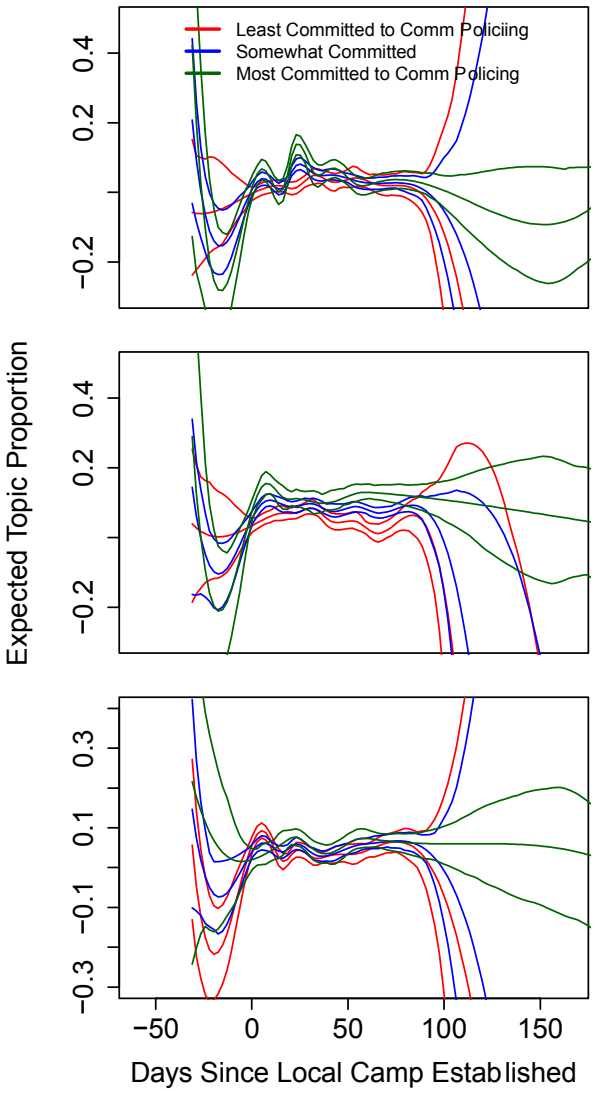


Note: Panel 1 (Top), Panel 2 (Middle), and Panel 3 (Bottom) show the predicted prevalence of Ordinance Enforcing, Group Arresting, Arresting Resisting Individuals in the corpus of text

units describing police-initiated events for each day of the Occupy movement. Each panel displays three lines predicting control performance prevalence for the highest, lowest, and middle value of the 'police experience with violent crime' variable interacted with time. See Methodological Appendix for equations used in estimation and prediction.

Figures 6.13 and 6.14, below, assess the extent to which community policing philosophies impact police control performances. Panel 1 of Figure 6.13 shows that departments most committed to community policing were also most likely to use Ordinance Enforcing to discourage protesters. They also steadily used Arrests (Panels 2 & 3), first primarily of groups, then both groups and individuals.

Figure 6.13 – Ordinance Enforcing, Group Arresting, Arresting Resisting Individuals by Commitment to Community Policing Interacted with Time

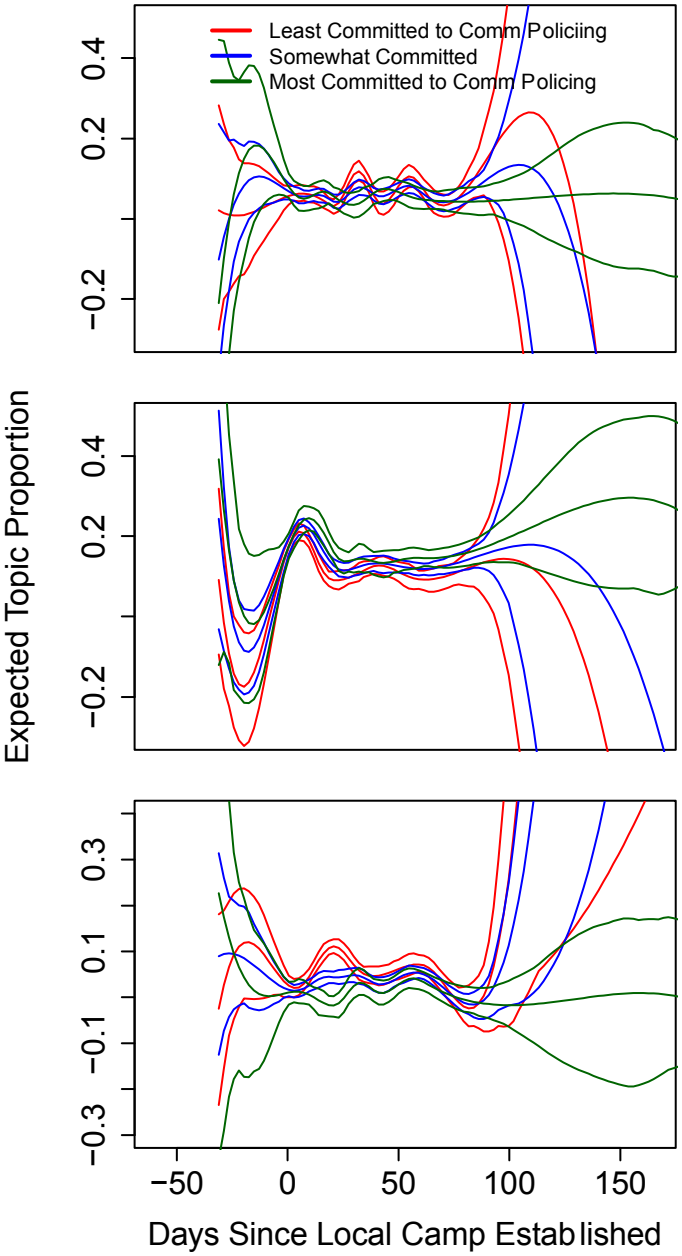


Note: Panel 1 (Top), Panel 2 (Middle), and Panel 3 (Bottom) show the predicted prevalence of Ordinance Enforcing, Group Arresting, Arresting Resisting Individuals in the corpus of text units describing police-initiated events for each day of the Occupy movement. Each panel displays three lines predicting control performance prevalence for the highest, lowest, and middle value of the 'community policing' variable interacted with time. See Methodological Appendix for equations used in estimation and prediction.

Departments most committed to community policing appear to have thoroughly avoided Raiding camps, too. While they peacefully Dismantled Camps as much (or more) as departments less committed to community policing, they gave fewer Deadline ultimatums, and seem to have managed protest by exerting pressure on individuals through Citations

and Arrests (Figure 6.13 Panels 1 and 2). Perhaps the orientation to community influenced departments' decisions to avoid direct confrontation with the local Occupy community and instead focus on enforcement actions against individuals and smaller groups of protesters.

Figure 6.14 – Deadline Enforcing, Dismantling Camps, and Violent Raiding by Commitment to Community Policing Interacted with Time



Note: Panel 1 (Top), Panel 2 (Middle), and Panel 3 (Bottom) show the predicted prevalence of Ordinance Enforcing, Group Arresting, Arresting Resisting Individuals in the corpus of text units describing police-initiated events for each day of the Occupy movement. Each panel displays three lines predicting control performance prevalence for the highest, lowest, and middle value of the 'community policing' variable interacted with time. See Methodological Appendix for equations used in estimation and prediction.

Discussion

Hypothesis 6.6 was confirmed. Police departments in cities that concentrate more power in fewer hands appear to have acted more decisively and eschewed "wait and see" strategies. They engaged Occupy campaigns more forcefully and sooner. They Arrested more people early in campaigns, and chose not to bother with the tactic of strategically hassling protesters through citations. Generally more active, cities with stronger executives controlling more power Dismantled camps early on, then Raided camps, then set Deadlines, then Raided any remaining camps in quicker succession than their counterparts. It appears that while U.S. Police Departments may enjoy more autonomy than their European Counterparts, this is less true for departments in cities where power is concentrated in the hands of fewer 'deciders.' Those cities kept departments busy wrestling with local campaigns.

Hypothesis 6.7 is largely confirmed. Police departments in cities that were undergoing an electoral transition were more likely to defer decisions about how to respond to the movement. They were a couple days later in attempting to Dismantle initial encampments (Figure 6.4 Panel 2) and less likely to engage in Raids until later in campaigns (Panel 3). Instead, these departments targeted individuals for Citation more frequently than their counterparts (Figure 6.3 Panel 1) avoiding the "in the job trouble" that might come with directly confronting Occupy campaigns too soon. While they may not have been more accommodating per se, as Hypothesis 6.9 states, police in politically unstable cities did seem to delay activity, probably to seek elite advice.

Hypothesis 6.8 is confirmed. When police in politically unstable cities did engage, they were significantly more likely to pick off protesters one-by-one with Individual Arrests (Figure 6.3 Panel 3). One (with an imagination) can almost picture the local party boss directing police to squash the movement and keep it off the front page.

Hypothesis 6.9 is confirmed. Police in smaller, more conservative districts showed no tolerance for Occupy campaigns at any point, likely reflecting the sensibilities of political elites. They enforced Ordinances with zeal, Arrested Groups at higher rates, and persistently Dismantled any tents set up by Occupiers. They never let camps gain a steady foothold in their communities.

Hypothesis 6.10 is confirmed. Though departments in cities with larger, liberal populations were more accommodating than their counterparts, when they did clash with protesters, perhaps because they felt threatened by large crowds *and* a large pool of reinforcements, police were more likely to use *control performances* that showed their concern for their own safety. They wore riot gear, fired projectiles from a distance, and hid behind riot shields.

Hypothesis 6.11 was mostly rejected. It was thought that cash-strapped departments might feel a sense of urgency about ending protest campaigns early in order to avoid the soaring costs of controlling protest. However, there is little evidence, here, to support such a hypothesis. The timing of Raiding and Camp Dismantling control performances (Panels 3 and 2 of Figure 6.8, respectively) were similar across cities with rich and poor police departments. If anything, richer departments set earlier eviction Deadlines and were more likely to Dismantle Camps (peaceably) over the course of campaigns (Panels 1 and 2, respectively).

Finding 6.a: Though it was not hypothesized, readers may be unsurprised to learn that cash-strapped departments were more likely to engage in the Enforcement of Ordinances (Panel 1 of Figure 6.7). It seems these departments, to some extent greater than their richer counterparts, viewed their local Occupy campaign as a revenue-generating opportunity.

Hypothesis 6.12 is confirmed. Police departments with fewer officers per capita, used the strategic element of surprise and initiative, perhaps motivated by the desire to avoid weeks of confrontation with an Occupation campaign. Like their better-staffed counterparts, many of these departments immediately Dismantled encampments before they could even fully establish themselves (Figure 6.10 Panel 2). They showed they were serious in the next weekend, too, Violently Raiding camps (Figure 6.10 Panel 3) at a time when better-staffed departments were just issuing Deadline ultimatums (Figure 6.10 Panel 1). Then, in the next

week, these departments issued a raft of tickets for Ordinance violations and issued their own Deadlines that they immediately enforced with another round of Raids.

Hypothesis 6.13 is confirmed. Departments with limited staff, though, were also more likely to Arrest Individuals one at a time, avoiding force-on-force battles they might be ill-suited to engage (Figure 6.9 Panel 3).

Hypothesis 6.14 is confirmed. Police departments with a philosophy of community policing were more likely to treat movements with care, in accordance with a conception of the encampments as communities and potential allies that could help police root out illegal/unwanted behavior. In accordance with their community orientation to local campaigns, these departments were less likely to Raid camps, and even less likely to force Deadlines on them.

Hypothesis 6.15 is confirmed. Police departments with a philosophy of community policing were more likely to control movements through Arrests of Individuals and Groups than their counterparts.

Hypothesis 6.16 is confirmed. Police in cities with higher violent crime rates treated Occupy campaigns as more of an annoyance than some great threat to public order. At least initially, they were more likely to react to their local Occupy campaign with a shrug (Figure 6.12 Panel 2) compared to other cities. They did not bother Enforcing pesky Ordinances compared to cities with less violent crime. And when they did deal with camps, they were most likely to start by issuing a Deadline ultimatum that they hardly enforced. When that failed, they issued a second Deadline (Figure 6.12 Panel 1) that they did enforce with Arrests of both Groups and Individuals (Figure 6.11 Panels 2&3). But, these departments seemed content to disperse Occupiers without much use of riot gear, tear gas, or the all the trappings of Violent Raids (Figure 6.12 Panel 3). In line with Hypothesis 6.16, officers who more regularly witness serious criminal activity, seemed not to view protesters as a major threat requiring counterinsurgency equipment.

Hypothesis 6.17 was not supported. Whether because Occupy movements were predominantly white, middle class phenomena, or because police officers' ethnicity matters little to their protest policing calculus, there was no evidence that the percentage of

nonwhite officers in a department affected their *control performances* in any appreciable way. In the interest of space visualizations of these models have been omitted.

These findings warrant significant discussion, the opening task of Chapter 7.

Chapter 7: Discussion and Conclusions

This dissertation has endeavored to answer the following questions about protest policing:

> *Do broad factors thought to influence the policing of protest – features of the governing context, police capacities, and police cultures – indeed influence police responses to protest?*
>
> *If these broad factors do influence the policing of protest, how do their effects vary over the course of protest campaigns?*

With results from the analyses of Chapter 6, we have some answers worth discussing further.

Political Opportunities' Effects on Protest Policing

Small Conservative Towns, Liberal Cities

Researchers have documented a number of ways in which political context, or the structure of political opportunities, affect both protest and the policing of protest. In democratic societies, where the political elites who (more or less) supervise police departments are beholden to voters, social movements often seek elite favor. Even when social movements challenge elites, those elites must consider the effect of their response on their future elections and current political alliances. It should come as little surprise, therefore, that so many scholars have recorded causal linkages between social movements' relative ideological affinity with elites and the subsequent policing of protest. When movements of the left make claims on political elites of the left, protest policing tends to be gentler than when similar movements target political elites on the political right (Fillieule 1997: 335-40; della Porta 1995; Geary 1985: chapter 7; Winter 1998). This dissertation finds, in line with these scholars (and confirming Hypothesis 6.9), that police in smaller, more conservative districts showed no tolerance for Occupy campaigns at any point, likely reflecting the sensibilities of political elites. They enforced ordinances with zeal, arrested groups at higher rates, strictly enforced deadlines and eviction orders, and persistently dismantled any tents set up by Occupiers.

However, authorities' willingness to accommodate potential ideological allies' camping on their City Hall lawns and blocking Traffic (Figure 5.12 Panel 1) was not infinite even in large, liberal cities. Whether elites were flexing their muscles to show their 'law and order' *bona fides* as Funk (1991) and della Porta (1999) might suggest, simply allowing police free reign as Earl and Soule (2006) might argue, or just facing larger, more radical Occupy campaigns; police in larger liberal cities were more likely to battle with protesters. They engaged protesters in more curfew disputes, arrested more of them during their *contentious gatherings* (Panels 2&3 of Fig. 5.12), arrested more individual protesters sooner, and violently raided camps more, and more frequently. Hypothesis 6.10, based on the broader hypothesis that *control performances* are often chosen based on a police departments' sense of threat to their control, predicted such an outcome.

Political Instability

The typical accommodation pattern noticeable when movements and elites share an ideological affinity is thought to be all the stronger when elections are upcoming or have just occurred. According to Tilly's conceptualization of political opportunity structures (outlined in Chapter 4), these are moments when elites are seeking allies, either voters or coalitions of activists and special interests to help them carry forward their agendas. If, indeed, political elites influence protest policing, *control performances* should be rather more accommodating during periods of political instability.

As across all cases, police departments in cities where elites were vying for support eventually shut down their local Occupy campaign. But police in these cities were slower to act than their counterparts, suggesting that they sought advice and consent from indecisive elites. They were a couple days later in attempting to dismantle initial encampments (Figure 6.4 Panel 2) and less likely to engage in raids until later in campaigns (Panel 3). Instead, these departments attempted to manage *campaigns* with deadlines (Fig. 6.4 Panel 1) and targeted individuals for citation and arrest more frequently than their counterparts (Figure 6.3 Panels 1&3 and Figure 5.20 Panel 3) avoiding the "in the job trouble" (Waddington 1998) that might come with directly confronting Occupy campaigns too soon.

Centers of Power

Scholars debate the impact of government form on *protest*. On the one hand, divided governments feature more access points for movements and more potential allies with power to make the changes activists seek. Thus, Tilly and Tarrow (2007) and others (Kitschelt 1986; Kriesi 1995) would likely hypothesize that Council and Commission style city governments, with their multiple power centers, provide more opportunities for movements than mayor-led governments. Others argue that more centers of power just equal more veto points, more obstacles to winning a clear victory (Huber, Ragin and Stephens 1993; Skocpol 1992; Amenta and Young 1999).

None yet, however, have tested the impact of divided government on protest *policing*. Convinced that more veto points are likely to result in confusion among political elites and police chiefs, this study has hypothesized that police in mayor-led cities with fewer centers of power would act more decisively against Occupy campaigns. Data bear out that prediction. Police departments in cities that concentrate power in fewer hands eschewed "wait and see" strategies. They engaged Occupy campaigns sooner. They arrested more people early in campaigns, and chose not to bother with the tactic of strategically hassling protesters through the enforcement of city ordinances. Generally more active, cities with stronger executives controlling more power dismantled camps early on, then raided camps, then set deadlines, then raided any remaining camps in quicker succession than their counterparts. It appears that while U.S. Police Departments may enjoy more autonomy than their European Counterparts, this is less true for departments in cities where power is concentrated in the hands of fewer 'deciders.' Those cities kept police departments busy wrestling with local campaigns.

Police Capacity's Effects on Protest Policing

Like any actor in any context, a police department's ability to deliver a particular *control performance* will likely depend on its capacity to deliver *control performances* in general. Some of this capacity comes down to knowledge and training (a variable difficult to measure (so far)), but much of it comes down to basic resources: personnel, and the budget to pay their wages. There has been very little research examining the effects of budget on

repression and protest policing (though see Ron 2000; and Boudreau 2005; 2009). Small N studies often take capacity for granted and just focus on the *control performances* carried out in their particular cases. Quantitative scholars usually focus on the large-scale trends in repression across decades and countries, not the allocation of *control performances* during particular cycles or episodes of contention. Earl and Soule (2006) and Earl, Soule, and McCarthy (2003), however, have included a simple measure of capacity, police budget per capita, in their models. They found that well-resource departments were less likely to engage in violence against protesters. This dissertation uses Earl et al.'s measure plus a second measuring the number of officers per citizen in a city.

Police Budgets

Results show, contrary to Hypothesis 6.11 that police budget had only a modest impact on most police *control performances*. It was thought that cash-strapped departments might feel a sense of urgency about ending protest campaigns early in order to avoid the soaring costs of controlling protest. However, the timing of raids and camp dismantling control performances (Panels 3 and 2 of Figure 6.8, respectively) were similar across cities with rich and poor police departments. If anything, richer departments set earlier eviction deadlines and were more likely to dismantle camps (peaceably) over the course of campaigns (Panels 1 and 2, respectively). Readers may be unsurprised to learn that cash-strapped departments were more likely to engage in the enforcement of ordinances (Panel 1 of Figure 6.7). These departments, to some extent greater than their richer counterparts, may have viewed their local Occupy campaign as a source of revenue. The greatest difference in *control performance* prevalence, by far, concerned police arresting individuals (Panel 3 of Fig. 6.7). Further research (or at least consultation with an informant from a police department) is needed, but perhaps better-resourced departments have personnel who assist in the generation of booking paperwork. Without such help, poorer departments may lack the capacity to pursue a strategy of movement control that relies on weakening movements by arresting their individual members.

Police Personnel

Results of models assessing the impact of departmental personnel per capita performed more to expectations. As predicted by a hypothesis (6.12) imagining police to be both strategic and mindful of threats to their control, departments with fewer officers per capita, used the strategic element of surprise and initiative to avoid prolonged confrontation with

their local Occupy campaign. Like their better-staffed counterparts, many of these departments immediately dismantled encampments before they could even fully establish themselves (Figure 6.10 Panel 2). But, they also showed they were serious in the next weekend, too, violently raiding camps (Figure 6.10 Panel 3) at a time when their better-staffed counterparts were just issuing deadline ultimatums (Figure 6.10 Panel 1). Then, in the next week, these departments issued a raft of tickets for Ordinance violations and their own Deadlines that they immediately enforced with another round of violent raids. Perhaps reflecting their relatively weak numbers, these departments were also less likely to engage in group arrests, instead picking off protesters one by one. It seems these department sought to avoid force-on-force battles except at times of their own choosing (Figure 6.9 Panel 3).

Police Culture's Effects on Protest Policing

The findings of this dissertation suggest that political elites – even in the U.S. context – have considerable influence over the protest policing decisions of their local police departments. But their input is not absolute. Department's bring a great deal of knowledge to the table when authorities make decisions about protest policing. Their "police knowledge," too, is infused into everything they do (della Porta 1998). Departments, like other organizations, develop a shared sense of their purpose, their values, and their best practices. They 'know' what is good policing and what is 'bad' policing and that 'knowledge' influences their practices in situations from street encounters with suspected drug dealers to clashes with protesters. While scholars have studied police cultures in general (Skolnick 1966; Wilson 1978; Lundman 1980) and note their influence on protest policing outcomes (Worden 1989; della Porta 1998), none yet have quantitatively and comparatively studied police culture's impact on protest policing as this dissertation does.

Community Policing Philosophy

Over the last two or three decades, a philosophy of 'community policing' has risen in popularity among police departments across the United States. Every major police professional organization endorses the approach, and the Bureau of Justice Statistics, an arm of the U.S. Federal Department of Justice, has recently added questions about departments' adherence to the philosophy to its periodic LEMAS survey. The survey is not the only tool the DOJ uses to prod departments to adopt the philosophy. Significant funding

opportunities are also tied to departments espousing of the creed. Community policing entails a number of methods to increase police integration into the communities they are to serve and protect. Police are encouraged to directly meet with neighborhood groups, civic institutions, schools, and religious institutions. They are urged to create open channels of communication through website that invite citizens to engage as police allies. And police are assigned to regular 'beats' where they are encouraged to leave their patrol cars, walk neighborhoods, and befriend the members of their local community.

This style of policing would seem to produce officers who are more communicative, less apt to see citizens as 'criminals' or 'problems,' and potentially more tolerant of free speech assemblies like Occupy campaigns. On the other hand, protests are rare. They do not especially resemble other situations in which officers may find themselves. So, it has also seemed plausible that a philosophy of community policing would have no impact whatsoever on protest policing *control performances*.

This dissertation has confirmed that, in fact, departments with a philosophy of community policing are more likely to treat movements with care. They appear, compared to their counterparts, to have conceived of encampments as communities and potential allies that could help police root out illegal/unwanted behavior. In accordance with their community orientation to local campaigns, these departments were much less likely to Raid camps, and even, less likely to force Deadlines upon them. Less likely to take down entire encampments, departments committed to community policing, instead focused their control performances on Individuals and small Groups, often by Enforcing city Ordinances.

View and Prioritization of Protest

This dissertation is the first piece of known scholarship to comparatively test the effects of police culture on protest policing. Della Porta has suggested (1998) that the way police view their work impacts their policing of protest, but this suggestion does not point one toward any particular hypothesis about which views result in which kinds of *control performances*. It was a conversation with a police officer (who also happens to hold a Ph.D. in sociology) that inspired Hypothesis 6.16 of this dissertation. This officer quipped that while officers in some rather sleepy towns and small cities might get rather animated about the Occupy movement – the biggest policing event in there area for over a decade – cops in larger, "harder" cities just saw it is as either a good opportunity for overtime pay or a mess that was going to keep them away from their families on weekends.

The officer appears to have had a point. Police in "harder" cities with higher violent crime rates treated Occupy campaigns as more of a nuisance than some great threat to public order. Their initial reaction to their local Occupy campaigns was mostly a non-reaction (Figure 6.12 Panel 2) compared to other cities. They did not bother Enforcing Ordinances against the campers. And when they did deal with camps, they were most likely to start by issuing a Deadline ultimatum that was mostly a bluff. When that failed, they issued a second Deadline (Figure 6.12 Panel 1) that they did enforce with Arrests of both Groups and Individuals (Figure 6.11 Panels 2&3). But, these departments seemed content to disperse Occupiers without much use of riot gear, tear gas, or the trappings of Violent Raids (Figure 6.12 Panel 3). In line with Hypothesis 6.16, officers who more regularly witness serious criminal activity, took a blasé approach the Occupy movement.

All of the hypotheses evaluated by this dissertation are listed and described in Table 7.1 below.

Table 7.1 Summary Evaluation of Hypotheses

Hypo-thesis #	Description	Variables	Confirmed?	Bears on Threat, Strategic, or Reactive Hypotheses?
4.1	Topic modeling will recover evidence of stable and well-rehearsed 'social movement' performances.	DVs: Rallies, Demonstrations, Weekday Marches	Yes	
4.2	Topic modeling will recover evidence of contentious performances unique to urban occupation campaigns including encampment in public spaces.	DVs: Encampment Activities, Weekend Gatherings, Curfew Disputes, City	Yes	

		Hall Targeting		
4.3	Topic modeling will recover evidence of contentious performances targeting banks, including bank transfer days and the blocking of entrances to banks.	DV: Bank Targeting	Yes	
4.4	Topic modeling will recover evidence of contentious performances blocking sidewalks and streets, and (b) these performances will happen more later in campaigns once performances have escalate to use more disruptive tactics.	DVs: Sidewalk Contestation, Traffic Battles	Yes But not 4.4b	
4.5	Topic modeling of performances through time will recover (a) evidence of an occupation campaign life course including (b) a sequence of activity beginning with camp establishment, performances of the traditional social movements' repertoire, and later more disruptive performances.		Yes But 4.5b was only mostly supported b/c of 4.4b. And there is enough variation in life courses to require that models testing Hypotheses 5.x – 6.x require POS	

			control variables.	
5.1	In cities featuring more independent centers of power, Occupy campaigns will be more active.	# centers of power	No	
5.2	In cities that have just experienced or will soon experience an election, campaigns will be more active, reflecting the general political activity concomitant with local political realignments.	Political instability	Yes	
5.3	In cities that have just experienced or will soon experience an election, elites will be more solicitous of Occupy campaigns, and as a consequence, protesters' performances will be less disruptive.	Political instability	Yes	
5.4	In cities where more Obama voters live, campaigns will be more active.	# of liberals	Yes	
5.5	In cities where more Obama voters live, campaigns will feature more performances designed for mass crowds, like marches and demonstrations.	# of liberals	Yes	
6.1	Topic modeling will recover evidence of control performances warning protesters to cease or modify	DV: Telling	Yes	

	their activities.			
6.2	Topic modeling will recover evidence of control performances aimed at 'strategic incapacitation,' the citation and ticketing of protesters for violation of minor city ordinances including overnight camping, violating curfews, jaywalking, etc.	DV: Ordinance Enforcing	Yes	
6.3	Topic modeling will recover evidence of control performances that seek to incapacitate and discourage movements by arresting individuals or small groups of protesters (even when police do not target the entire campaign for arrest.)	DVs: Group Arresting, Arresting Resisting Individuals	Yes	
6.4	Topic modeling will recover evidence of control performances that seek to peacefully close down encampments	DV: Dismantling Camps	Yes	
6.5	Topic modeling will recover evidence of control performances that seek to close down encampments through large quantities of arrests, police shows of force, and the use of "less lethal"	DV: Violently Raiding	Yes	

	weapons.			
6.6	Police performances will be more ambivalent to campaigns in cities where elites are fractured across many power centers.	# centers of power	Yes	Strategic, responding to elites
6.7	Police will be more accommodating to movements where electoral instability makes elites accommodating to voters.	Political instability	Yes	Strategic, responding to elites
6.8	Police will use more Individual Arrests to keep their control performances out of the headlines.	Political Instability	Yes	Strategic, responding to elites
6.9	Police performances will reflect their beliefs about the popularity of the movement with elected officials' constituencies. Departments in smaller more conservative towns will be less accommodating to Occupy campaigns.	# of liberals	Yes	Strategic, responding to elites
6.10	Police performances will reflect a sense of threat from larger crowds in liberal cities	# of liberals	Yes	Threatened
6.11	Police with smaller budgets are more likely to see long-lasting campaigns as a threat to their budgets, and may be more likely to shut down those campaigns (by any means necessary) sooner.	Police budget per capita	No	Threatened

6.12	Like departments with smaller budgets, departments with fewer officers per capita are more likely to see long-lasting campaigns as a threat to their workforce capacity, and may be more likely to shut down those campaigns (by any means necessary) sooner.	Police per capita	Yes	Threatened
6.13	Since these departments may have a general sense of being under-powered, they may prefer Arresting Individuals as opposed to performing Group Arrests.	Police per capita	Yes	Threatened
6.14	Police in departments committed to community policing will be more accommodating to movements in general.	Community Policing	Yes	
6.15	Since a tenant of community policing involves avoiding group punishments for individual actions, departments with this philosophy are likely to apply control to individuals more often than to camps.	Community Policing	Yes	
6.16	Departments in cities with high violent crime rates are less likely to view protesters as a major threat or priority, and therefore are likely to respond slower and with	Violent Crime	Yes	Threatened

	more accommodation.			
6.17	Departments with more non-white officers are generally less-likely to take a hardline against social justice movements, and so will be more accommodating to Occupy campaigns.	Nonwhite Officers	No	

Police-Centered Theory: Threatened? Reactive or Strategic?

Answers to the key questions opening this chapter bear on this dissertation's second large goal: to clarify police-centered theories of protest policing. The going theory of protest policing in the American context suggests (1) that police responses to protest are substantially motivated by their (sometimes inaccurate) sense of the threat protesters pose to their officers, and particularly to their control over situations. This 'Threatened Police Hypothesis' is evaluated by Hypotheses 6.10 – 6.13, and 6.16, above.

American sociological theory on the repression of movements also currently advances a second hypothesis (2), the 'Reactive Police Hypothesis,' suggesting that American police are autonomous from political elites and choose their *control performances* based solely on the their assessment of threats from protesters. This dissertation advances a third hypothesis (3), the 'Strategic Police Hypothesis.' This alternative suggests that police act strategically when faced with protest. They consider the elected officials who supervise them, take stock of their own department's resources, and act in line with their department's culture as they seek to limit the risk of disorder posed by protesters. They even take initiative at times. Hypotheses 6.6 – 6.9 of this dissertation offer crucial tests of the broader 'Reactive' and 'Strategic' hypotheses.

Crucial Test Results: Police are Strategic

If a strong version of the Reactive Hypothesis is true, the prevalence of police *control performances* should not vary across variables describing the *political opportunity structures* of the cities in which they are embedded. When movements in different cities behave similarly, police from those different cities should behave similarly in their response. If the 'Strategic Police' Hypothesis is correct, variations in the prevalence *and* timing of police *control performances* should be rather prominent in the data, a result of police taking decisions about protest policing in consideration of elites' concerns and their own sense (based on their resources) of the best moments to enact their *performances.*

As readers have seen in Chapter 6 and in the Table 7.1 above, the confirmation of Hypotheses 6.6 – 6.9, which test the effects of *political opportunity structure* variables on the prevalence of police *control performances*, thoroughly discredit the 'Reactive Police' Hypothesis. Police departments' *control performances* varied in prevalence and timing in ways that the broader literature on repression and protest policing (outside of the U.S.) would have predicted. Police waited on elites to decide their next steps (Hypothesis 6.6). In cities where elites were either up for election or just settling into new political coalitions, police were relatively accommodating campaigns (Hypothesis 6.7) and avoided the sorts of large battles likely to make headlines (Hypothesis 6.8). And in more conservative towns, police showed the leftists campers, from Day One, that they were unwanted by authorities (Hypothesis 6.9).

Strategic and Threatened

That police act strategically, does not imply that they always act without assessing the threats protesters pose to their officers. Police in large, liberal cities seem to have taken stock of the massive crowds that assembled in their public spaces, and then chosen, very early on, to beat back the movement with beanbag rounds, tear gas, and rubber bullets (Hypothesis 6.10). Departments that were relatively understaffed, too, refused to take any chances with their local Occupy campaigns. They also took strategic initiative, mustering all their forces to Raid camps early in campaigns (Hypothesis 6.12). These patterns of *control performances* look very much like a strategy of 'deterrence' that scholars have identified since 1966 (Feierabend & Feierabend), but that Earl and Soule have lately (2010) attempted to explain away. Understaffed departments also fought a sort of guerrilla war on local encampments, picking off protesters one-by-one with Individual Arrests (Hypothesis 6.13) rather than controlling them through larger Group Arrests that would require more

personnel over time. Such *performances* offer clear evidence of police acting strategically given their assessments that protesters threaten their control.

Earl and Soule's (2006) astute intuition that police carefully attend to threats to their control is brought into vivid relief by the testing of Hypothesis 6.16. If della Porta (1998) and Earl and Soule (2006) are right, not only should "police knowledge" affect the way officers perform their work, it should affect the way they assess threats to their control. That is exactly what a test of Hypothesis 6.16 finds. Police in cities with high crime rates, where threats to officers and the public are a regular phenomena, took a very nonchalant approach to their local Occupy campaigns. They viewed them as a nuisance perhaps, but not as a major threat to be met with riot gear and pepper spray. Compared to their counterparts, these departments spent far less time and energy on Occupy movements.

Political Opportunity Structures *not* Occupy Campaign Activities

Critical readers may harbor nagging doubts about findings showing that political opportunity structures affect police behavior. It has been a key contention of the American social movements repression literature that American police departments are insulated from political elites. By (a strong version of) this theory, the only way political opportunity structures could affect *control performances* is if they do so by encouraging or discouraging protester *contentious performances* that arouse reactive protest policing. This concern should be alleviated by the fact that all of the models testing the effects of political opportunity structure variables on the prevalence of police *control performances* included other political opportunity variables as statistical controls. Nonetheless, to satisfy any residual uncertainties, and to help provide food for further thought and discussion, I provide side-by-side visualizations of the effects of POS on both protester and police *performances* below. Readers have seen all of these plots before, and they retain their labels so that readers may cross-reference them in their original context, too.

Here, the critics' task is to note differences in protester's *contentious performances*, depicted in columns 1 and 2 of the following pages, and venture plausible reasons why

those differences caused the differences in police *control performances* depicted in column 3 of each of the following pages.

Centers of Power

First, we observe whether and how the number of power centers in a city may have affected protesters' activities in a way that explains differences in protest policing. Readers will note three significant, if small, differences in the prevalence of protesters' *contentious performances.* Protesters in cities with *more* power centers (with Council or Commission style governments that may also have been a state capitol) engaged in more Weekend Gatherings in the first week of their campaigns, and more Encampment Activities in their second week, than cities led by Mayors. They also appear to have experienced more spats with police over Sidewalks. Now, the question asked throughout this section is: do these differences explain differences in the prevalence of police *control performances?*

Recall from the discussion above, and observe, that what most distinguished police *control performances* across these cities was the decisiveness of police action in cities with fewer centers of power. Mayor led cities were quicker to Dismantle camps, quicker to Enforce Deadlines, and quicker to control Occupy campaigns through Group Arrests and Individual Arrests. It seems the findings discussed above hold up to closer critical analysis. Or maybe readers can venture a plausible account for why protesters' increased Weekend Gatherings, Encampment Activities, and Sidewalk Contests would slow the decision-making of cities with more power centers.

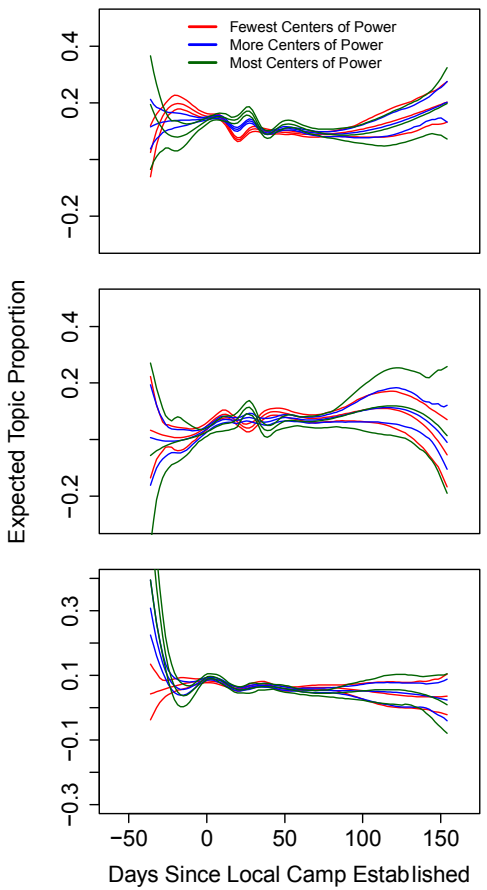


Figure 5.13 – Weekend Gatherings, Encampment Activities, Weekday Marches

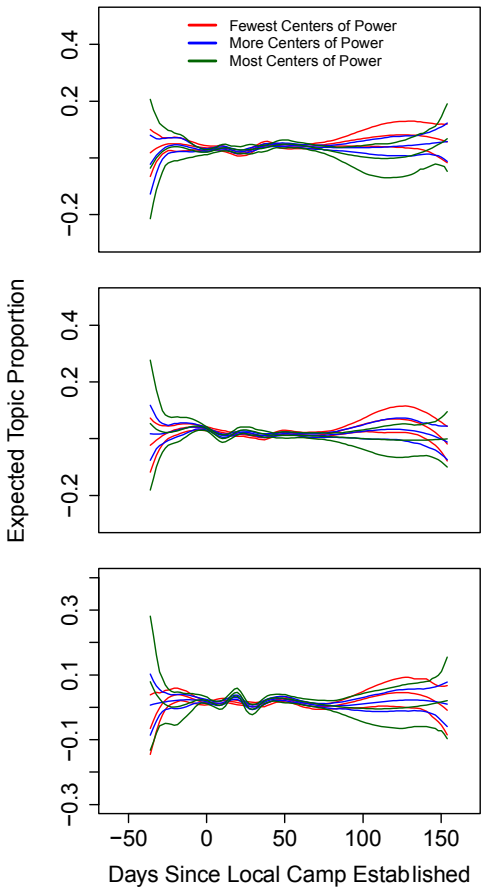


Figure 5.14 – Rallies, Demonstrations, and Labor Alliances

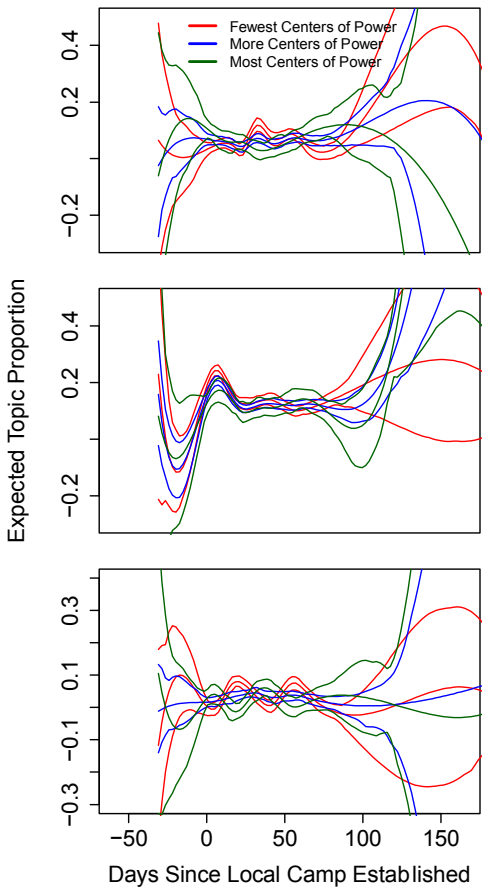


Figure 6.2 – Deadline Enforcing, Dismantling Camps, and Violent Raiding

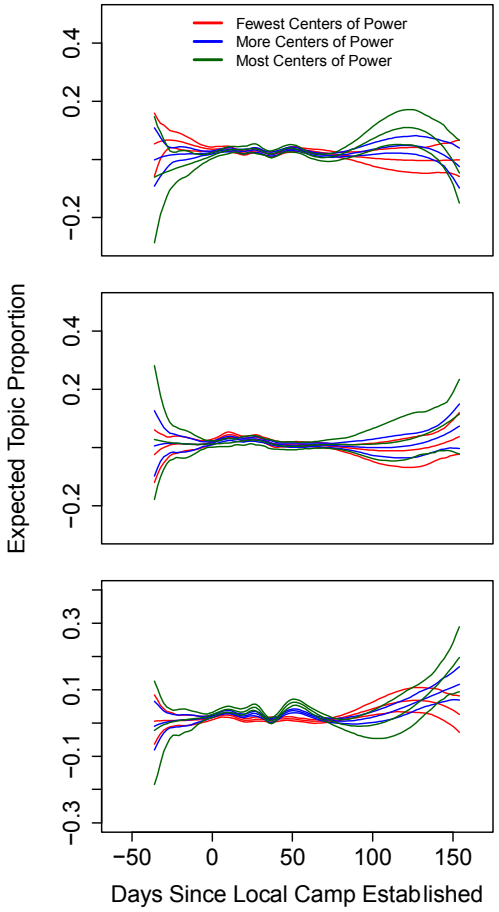


Figure 5.15 – City Hall Targeting, Bank Targeting, and Sidewalk Contestation

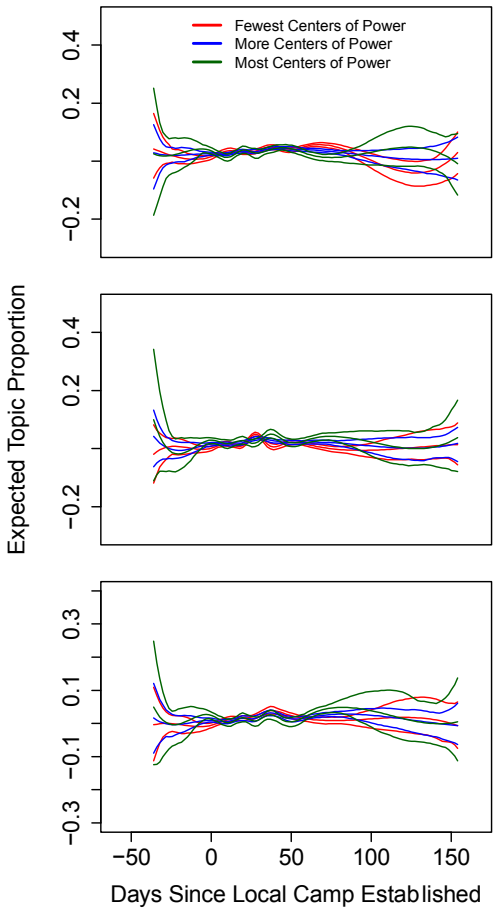


Figure 5.16 – Traffic Battles, Curfew Disputes, Arrests

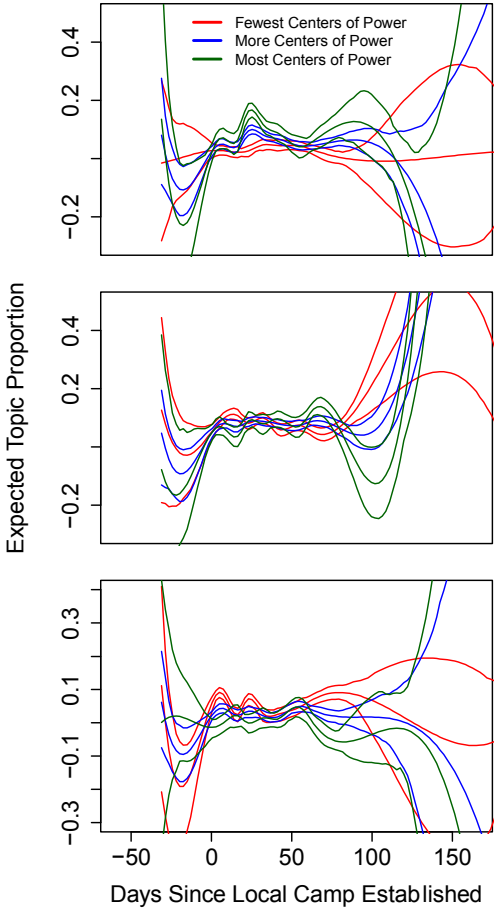


Figure 6.1 – Ordinance Enforcing, Group Arresting, Arresting Resisting Individuals

Political Instability

Continuing to double-check this dissertation's findings, readers will observe plots showing the effects of political stability on the prevalence of protester and police *performances* below. The most significant difference in protester activity across politically stable and unstable cities concerns Encampment Activities. As a general theory of political opportunity structures would predict, Occupy Encampments were more Active in cities that were undergoing a political transition. This increased activity aligns with the notion that elected officials tend to be more accommodating of movements around the time activists are taking to the voting booth. Indeed, as discussed above, political elites in politically unstable cities were slightly slower to act than their counterparts. They were a couple days later in attempting to Dismantle initial encampments (Figure 6.4 Panel 2) and less likely to engage in Raids until later in campaigns (Panel 3). Instead, these departments attempted to manage *campaigns* with Deadlines (Fig. 6.4 Panel 1) and targeted Individuals for Citation and Arrest more frequently than their counterparts in politically stable cities (Figure 6.3 Panels 1&3 and Figure 5.20 Panel 3).

These findings suggest *not* that protester's increased Encampment Activities caused unstable cities' relative accommodation of the movement, but that elites' accommodation of the movement allowed Occupy campaigns in politically unstable cities to engage in more Encampment Activities. Movement campaigns in unstable cities appear also to have been *slightly* more likely to target City Halls and Banks and to draw Arrests during their events. None of these differences would *explain* the differences in police behavior. The one, final difference concerns Curfew Disputes. It appears that in trying to avoid the large-scale Raids and confrontations that police in politically unstable cities eventually brought upon protesters, police Curfew Disputes evolved into Ordinance Enforcement in week 4, just as police in more stable cities were being to spar with protesters about Curfew orders. Especially since this difference concerns a *performance* in which police and protesters were engaged, there does not seem to be a way that it *explains* the pattern by which police in politically unstable cities seemed to delay high-profile efforts to shut down encampments.

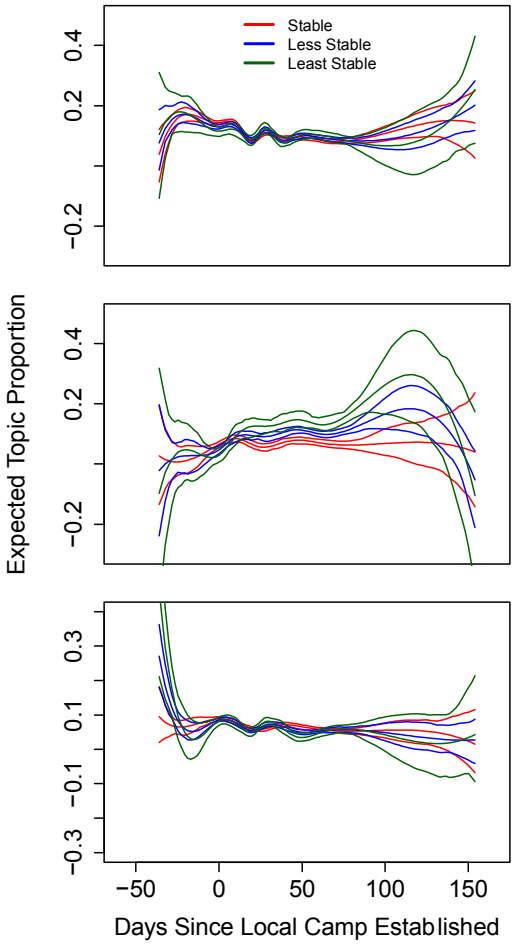


Figure 5.17 – Weekend Gatherings, Encampment Activities, Weekday Marches

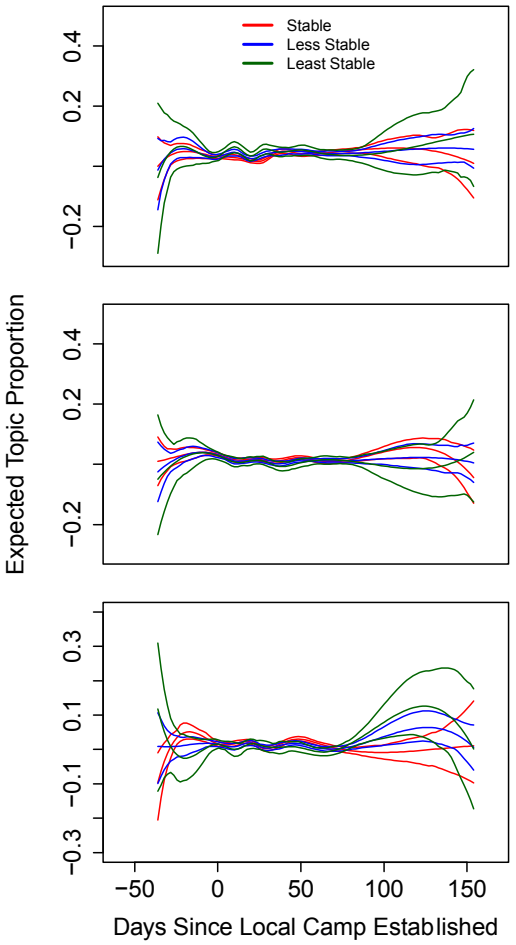


Figure 5.18 – Rallies, Demonstrations, and Labor Alliances

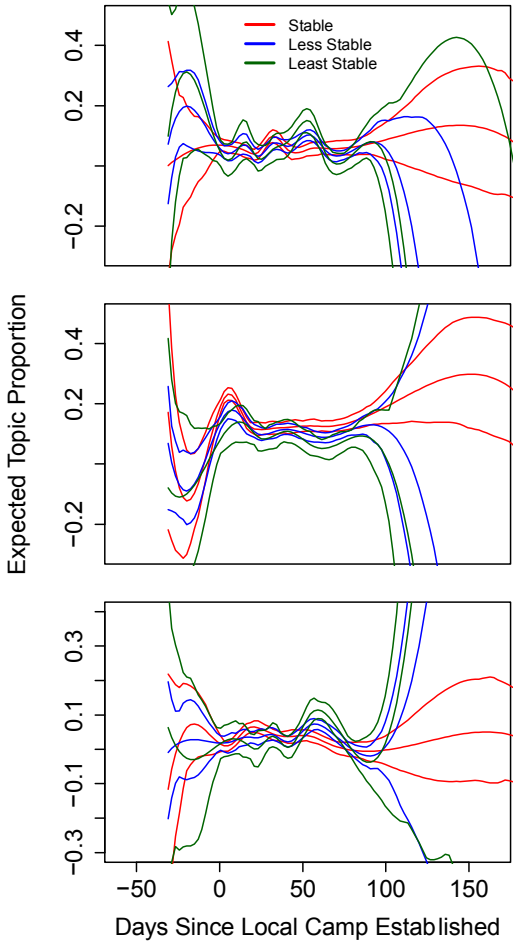


Figure 6.4 – Deadline Enforcing, Dismantling Camps, and Violent Raiding

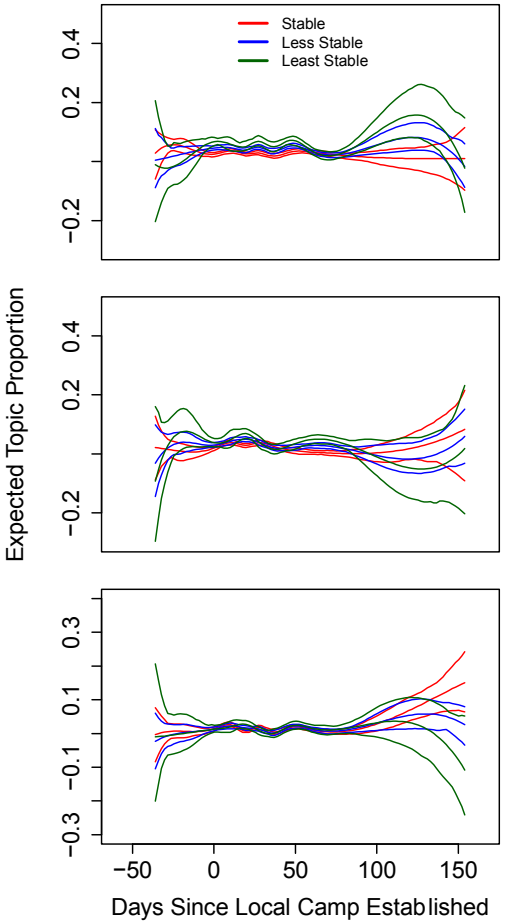


Figure 5.19– City Hall Targeting, Bank Targeting, and Sidewalk Contestation

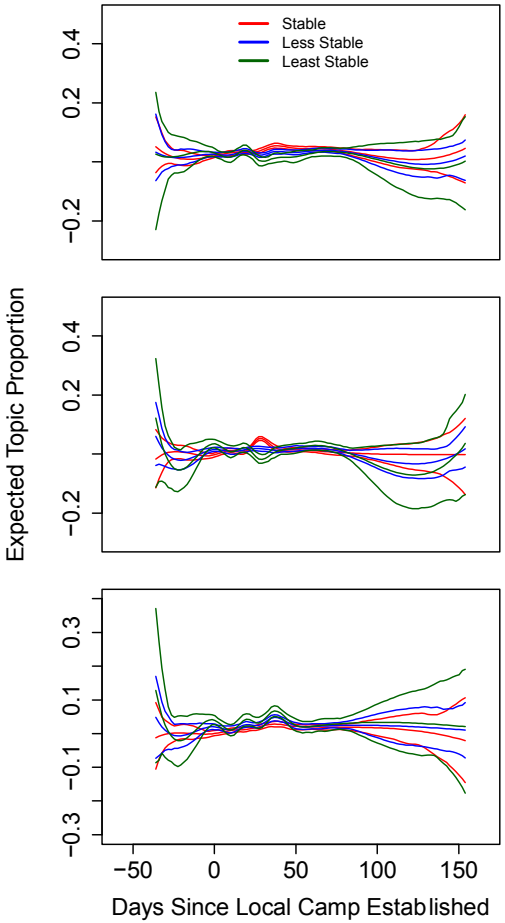


Figure 5.20 – Traffic Battles, Curfew Disputes, Arrests

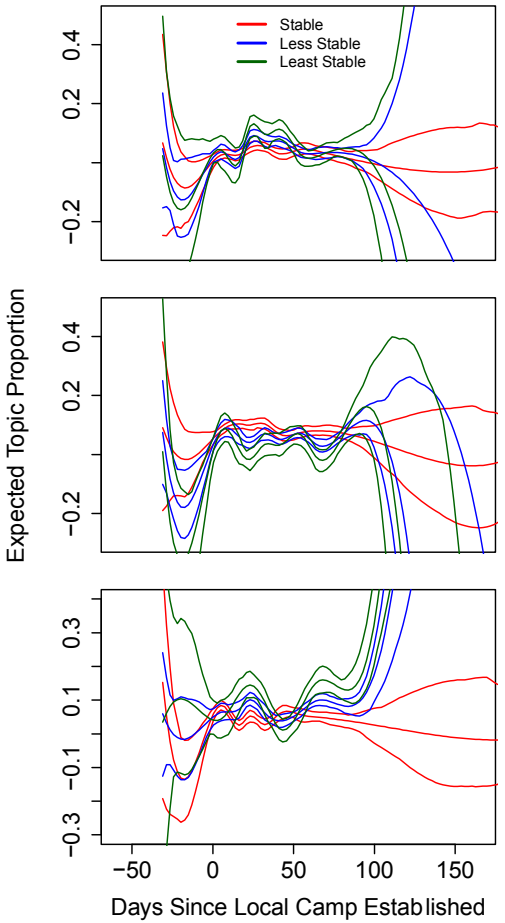


Figure 6.3 – Ordinance Enforcing, Group Arresting, Arresting Resisting Individuals

Size of Liberal Population

Finally, turning to the last set of plots, below, we explore whether differences in protesters' *contentious performances* across cities ranging from small and relatively conservative to large and liberal may have invited differing *control performances* from police. First of all, consider that across these cities there are no considerable differences in the prevalence of pure, protester-led *contentious performances* – those *performances* that do not include police. Large cities, perhaps, had more Labor support. Otherwise, the differences concern Traffic Battles, Curfew Disputes, and Arrests during protester-initiated *contentious gatherings*. The question becomes: did these shared protester/police *performances* cause the early use of Raids and Individual Arrests by police in larger, more liberal cities?

A close inspection shows that the first high peak of the Violent Raid *performance* occurs days before the first peak of the Traffic Battle *performance*. This suggests that police already had a plan to shut down camps, and that protesters' engagement in Traffic Battles may have given them the evidence they needed to justify camp closures. Or, perhaps, police reacted with a hair trigger to the very first Traffic Battles they experienced during the campaign. This dissertation does not wish to make the mistake of assuming that one party or another provoked a *performance* like a Traffic Battle. But whoever started those individual episodes, police clearly went on the offensive, Arresting many more individuals during the first week of their local campaign, foregoing Deadline warnings, engaging in increased Arrests during *contentious gatherings,* enacting Curfews, and Enforcing Ordinances quite early in *campaigns*. Though a test of this hypothesis will have to wait for a future study, the quick response of police in large, liberal cities may suggest that police were already wary of local activists *before* protesters even began pitching their tents.

By this theory, a strong police reaction to a campaign should not be taken as evidence that the campaign *caused* that police reaction. Indeed, again, there is little discernable difference between the *contentious performances* of campaigns in small-and-conservative vs. large-and-liberal cities. Moreover, there is nothing in any of the *performances* occurring during Occupy-initiated *contentious gatherings* that explains the *control performances* of police in smaller, less liberal cities and towns. Those departments chose to Enforce Deadlines and peacefully Dismantle Camps at a much higher rate. And they opted for Group Arresting, and Ordinance Enforcing at a much higher rate. These differences appear to reflect a strategy of driving Occupy campers away with steady, unwelcoming pressure.

Of the three measures of political opportunity structures, the number of liberals is the most likely to affect both campaign *performances* and police *control performances*. With a larger, liberal population, campaigns have more potential supporters to draw on, so they often bring larger crowds that are more likely to trigger a police sense of insecurity. However, if police were only responding to threat, police in smaller cities and towns with a more conservative population would have little reason to be so inhospitable to campaigns in their own way. Their intolerance for smaller, less threatening campaigns offers support to the notion that police consider the ideological affinities between movements and elites when making enforcement decisions. One wonders, for instance, if these same police are so inhospitable to more conservative movements – a question for future study.

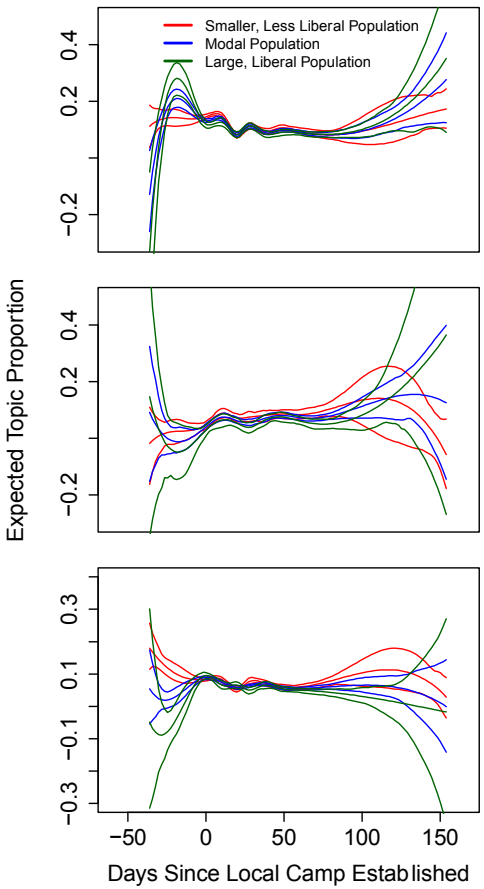


Figure 5.9 – Weekend Gatherings, Encampment Activities, Weekday Marches

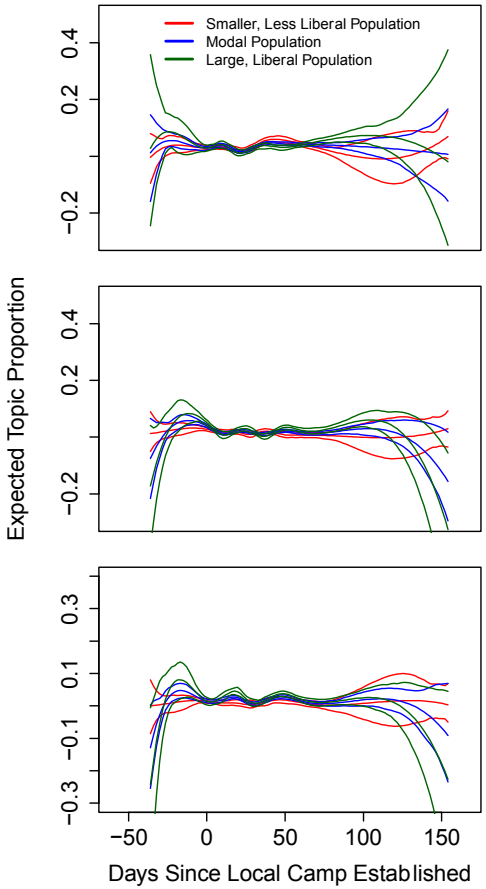


Figure 5.10 – Rallies, Demonstrations, and Labor Alliances

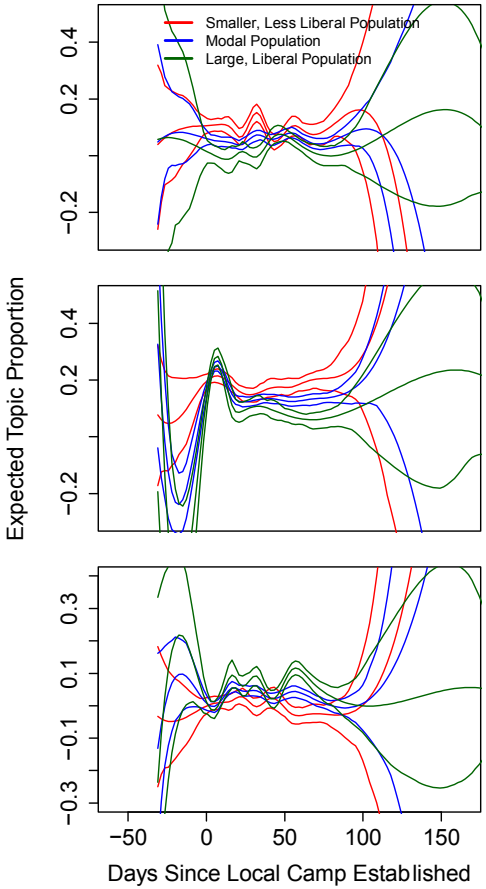


Figure 6.6 – Deadline Enforcing, Dismantling Camps, and Violent Raiding

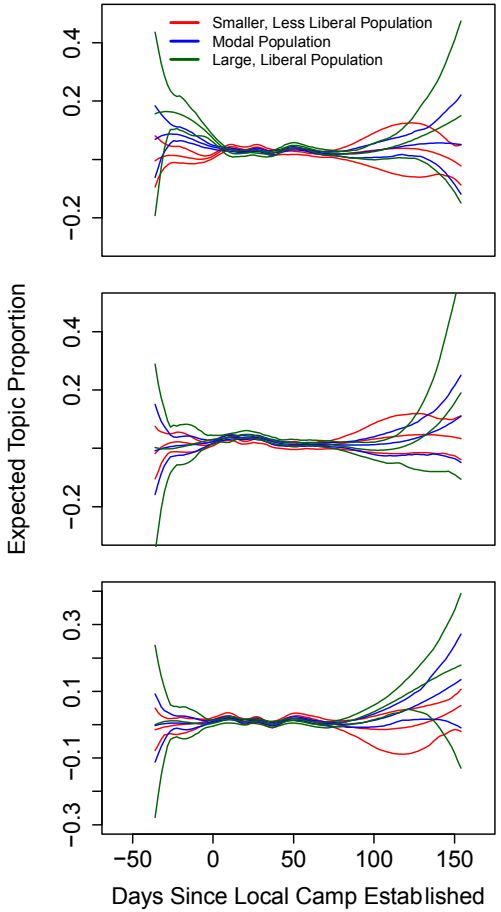


Figure 5.11 – City Hall Targeting, Bank Targeting, and Sidewalk Contestation

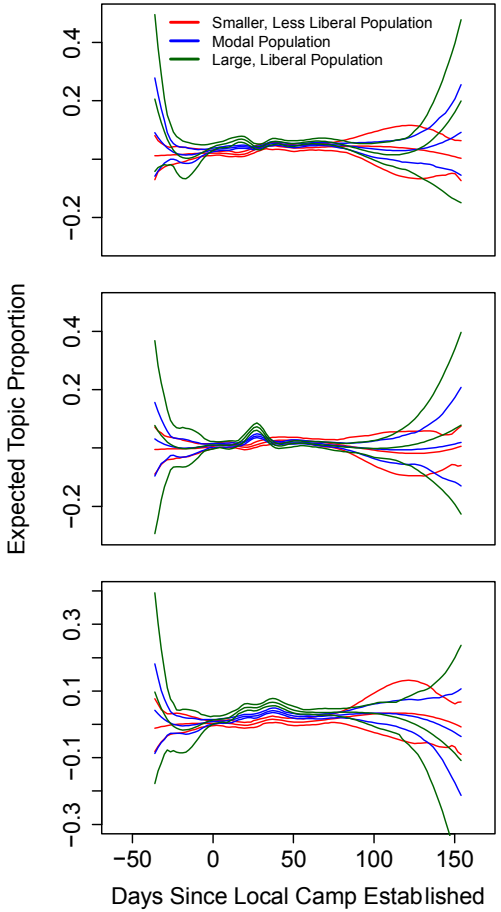


Figure 5.12 – Traffic Battles, Curfew Disputes, Arrests

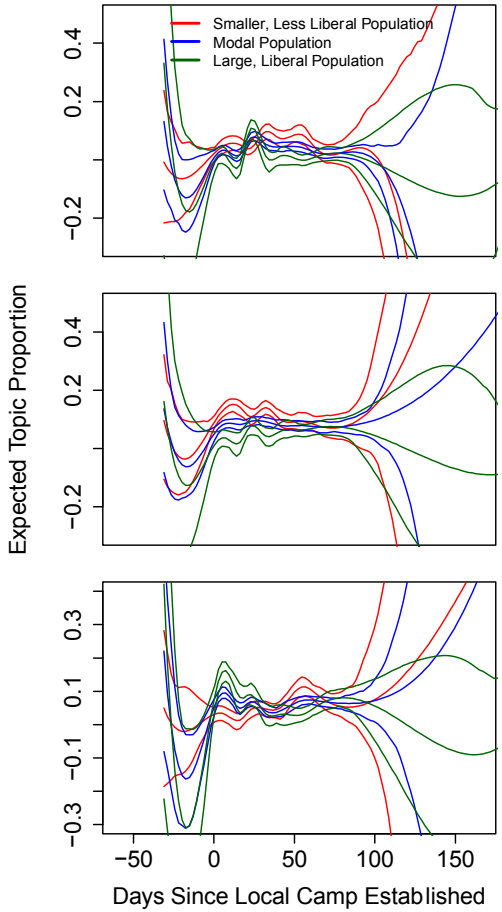


Figure 6.5 – Ordinance Enforcing, Group Arresting, Arresting Resisting Individuals

Overall, there is no compelling case to overturn the conclusion that political opportunity structures independently shape police *control performances.* The number of power centers, in particular, seems to affect decision-making around protest policing. And political instability leads to more accommodating responses, or at least responses that attempt to delay a showdown with protesters. More research is needed to understand how the pool of potential government and movement supporters affects police *control performances*. Current results are crosscutting. One might expect that police were more accommodating to liberal movements in liberal cities, but whether because of difficult histories with past movements, a sense of greater danger from larger crowds, or some other reason, they displayed extraordinarily little patience with Occupy camps. On the other hand, as expected, smaller conservative cities and towns were also unaccommodating to camps in their own way, perhaps because they went against the grain of prevailing political ideologies.

In any case, the relatively large differences in the prevalence and timing of police *control performances* when compared to variations in *contentious performances* (The scales on all plots are identical.) also suggests that differences in policing are *not completely* tied to protesters' activities. Differences in protest policing that emerge as a consequence of police culture underline that point. The sum total of these analyses thoroughly repudiate a strong version of the Reactive Police hypothesis.

Readers might be tempted to draw the conclusion not only that police behaved strategically during the Occupy movement, but also that they have behaved strategically throughout the last several decades of American protest – that Earl, Soule, Davenport, and McCarthy have been wrong all along. This dissertation does not go so far. Its data only concern Occupy movements in the Fall of 2011. However, it is worth noting that the data analyzed by Earl, Soule, Davenport and McCarthy are fundamentally incapable of discovering indicators of strategic action that are apparent in studies (until now only small-N studies) of protest *campaigns.* Those indicators, of delayed action or strategic initiative-taking, cannot be observed when researcher treat each *contentious gathering* as its own phenomenon, unconnected to any larger string of police and protester interaction. Future research using the data of Earl and her colleagues might seek to identify and analyze *campaigns* of protester *contentious gatherings* and include more measures of *political opportunity structure.*

Such an approach would clarify whether or not police during the 1960s, '70s, and '80s were merely reacting to protesters or also behaving strategically as this dissertation has shown.

Conclusions

This dissertation concludes with an evaluation of its contributions to various literatures and the public.

Since the substantive contributions of the foregoing research have only been possible thanks to significant advancements in the methods of event analysis and text analysis, we begin there. For decades, quantitative researchers have hand-coded event data from newspapers. Resulting databases, describing variables solely at the event level, have been used to discern long-term and cross-country trends in the prevalence of social movement event-types (marches, vigils, kidnappings, etc.) and government efforts to repress them. Seeking to understand, in greater detail, the *actions* occurring within those *events,* a smaller group of determined researchers have also coded the subjects, verbs, and objects (who does what to whom) appearing in accounts of events. The work is painstaking, requiring many years and/or large teams of research assistants, but the payoff is great, allowing scholars to understand how certain actions (e.g. walking, chanting) cohere into some events (e.g. marches), while other actions (e.g. lighting candles, saying prayers) cohere into different events (e.g. candlelight vigils). This approach has allowed scholars like Charles Tilly to uncover and explain the myriad small innovations in *contentious performances* that other scholars viewing events could only observe as high-level trends that, perhaps, required them to update their event categorization schemes.

This dissertation project has automated significant portions of this painstaking workflow, and goes further. Rather than hand-coding news articles for all variables about Occupy events, this dissertation's research team merely labeled portions of news text referring to a single *contentious gathering* or *police-initiated event*. Then, thanks to recent advancements in computational linguistics, it was possible to

extract subject-verb-object data from these event-level text units automatically. While the first step required a year of effort, the second requires only a few hours (once one is acquainted with the commands that perform the SVO extraction). With some (also automated) data cleaning (allowing SVO-amplification of text units, event date assignments, and actor normalizations, as described in Chapter 3), these data can then be used to automatically (via LDA, a sort of clustering algorithm that performs topic modeling) induce coherent *performances* constructed of SVO *actions*. Since these *performances* and *actions* are organized by the events, cities, and dates in which and on which they occurred, researchers can then ascertain how city-level variables (like those describing political opportunity structures and police departments) affect the prevalence of *performances* through time over the course of *campaigns.* To some researchers who have been seeking a way to record data on all of these nested units of analysis (actions, performances, events, campaigns, cities) for some time, Hanspeter Kriesi (2009) among them, it is hoped that these methods and design will be welcomed. It is also hoped that the larger research community will pick up these tools to study campaigns describing event and social movement phenomena beyond the Occupy movement.

This dissertation also makes other, substantive contributions to the social movements literature. The efforts of Chapter 4 and 5, discerning the extent to which Occupy campaigns unfolded according to a common life course, revive a fairly dormant ambition of the subfield. Since Blumer's abandoned efforts in the last century, scholars have called for the division of social movement analysis by common social movement stages (Shultziner 2014), but few have made significant advancements in this area, none performing quantitative research. This dissertation confirms Shultziner's assertion that the active stage of social movements – between their origins and outcomes – can be fruitfully studies in terms of political opportunity structures.

More importantly, perhaps, this dissertation upgrades the ontological status of police repression from a static element of political opportunity structure to dynamic *control performances* to be studied in interaction with *contentious performances*. This update has been long overdue – the stubbornness of the static conceptualization of repression owing, no doubt, to the inadequacy of previous methods. It has been difficult enough to collect data on what *social movements* are up to, often requiring many years of hand-coding by many people. Researchers have

rarely had the time and funding to collect similarly detailed data on authorities' activities. Instead, they have just conceived of repression as a static phenomenon.

To the American social movements literature, in particular, this dissertation offers one final contribution: an update to theories of protest policing. Until now the final word on American protest policing has been that police behavior is mostly determined *in reaction* to protesters. If the police response to a protest is rather anodyne, it is because protesters are behaving appropriately. If police are violent, it is because police felt threatened by protesters. Whether this theory of American protest policing is correct or not, it does not seem to square with recent experiences.

This dissertation clearly shows, in quantitative comparative fashion, what many observers and participants in police and protester interactions (including police) have observed: that police are strategic. They are clever. They play cat and mouse. They wait until the right moment. They consider their political allies.

This dissertation makes more general contributions to the more general social science (political science and sociology) literatures on repression and protest policing as well. It hopes not only to show clear evidence of police strategizing, but also to encourage more research in this area. Perhaps this work will offer some relief to a research community that has long been frustrated by apparently contradictory findings. It clarifies that many of the discordant findings in the literature can be resolved by observing police and protester interactions through time. Yes, police of a single city facing a single Occupy campaign forego Raiding camps. And, yes, those same police facing that same Occupy campaign *also* Violently Raid camps with brutal force. There is no contradiction. They just choose one course of action early in a campaign and the other later in the same campaign.

The reasons for these shifting patterns of activity need not mystify researchers either. They depend on police departments' assessments of threats to their control posed by protesters (assessments colored by their understandings of their jobs), and their assessments of political elites' wishes. They act strategically in ways that researches can further investigate as long as they have data able to reveal how parties' *performances* change over the course of *campaigns*.

Finally, to the public, this dissertation offers a framework for further discussion about the state of protest policing in our society. The data are in. They are not distorted by pundits. They are not cherry-picked by politicians. They are as inductively-generated as can be, as clean as can be. And they show that police and political elites *do* make choices when facing crowds. They strategize. They choose to accommodate sometimes. They choose to fire tear gas and rubber bullets at other times. And they make those choices based on political expediency, based on their sense of threat (real or imagined), and based on what police work means to them. It is up to the American public to decide what of this is okay, what of this counts as law and order, as democracy, as brutality, and what, if anything, should be done to encourage different patterns of police and protester interactions in the future.

Works Cited

Alimi, Eitan Y., and David S. Meyer. 2011. “Seasons of Change: Arab Spring and Political Opportunities.” *Swiss Political Science Review* 17 (4): 475–79.

Alimi, Eitan. 2009. “Mobilizing Under the Gun: Theorizing Political Opportunity Structure in a Highly Repressive Setting.” *Mobilization: An International Quarterly* 14 (2): 219–37.

Almeida, Paul D. 2008. *Waves of Protest: Popular Struggle in El Salvador, 1925-2005*. Vol. 29. U of Minnesota Press. https://books.google.com/books?hl=en&lr=&id=BDkzorCHZhYC&oi=fnd&pg=PR7&dq=almeida+2008+protest&ots=RkW1CWehpo&sig=ePqObKWQUwqfD2nifO1gaTX-j0M.

Amenta, Edwin, and Michael P. Young. 1999. “Democratic States and Social Movements: Theoretical Arguments and Hypotheses.” *SOCIAL PROBLEMS-NEW YORK-* 46: 153–68.

Amenta, Edwin, and Neal Caren. 2004. “The Legislative, Organizational, and Beneficiary Consequences of State-oriented Challengers.” *The Blackwell Companion to Social Movements*, 461–88.

Amenta, Edwin, and Yvonne Zylan. 1991. “It Happened Here: Political Opportunity, the New Institutionalism, and the Townsend Movement.” *American Sociological Review*, 250–65.

Association, International City/County Management, and others. 2010. “The Municipal Year Book 2010.” Washington, DC.

Balch, Robert W. 1972. “The Police Personality: Fact or Fiction?” *The Journal of Criminal Law, Criminology, and Police Science*, 106–19.

Barranco, José, and Dominique Wisler. 1999. “Validity and Systematicity of Newspaper Data in Event Analysis.” *European Sociological Review* 15 (3): 301–22.

Beck, Elwood M., and Stewart E. Tolnay. 1990. “The Killing Fields of the Deep South: The Market for Cotton and the Lynching of Blacks, 1882-1930.” *American Sociological Review*, 526–39.

Bessel, Richard, and Clive Emsley. 2000. *Patterns of Provocation: Police and Public Disorder*. Berghahn Books.

Blei, David M., Andrew Y. Ng, and Michael I. Jordan. 2003. “Latent Dirichlet Allocation.” *The Journal of Machine Learning Research* 3: 993–1022.

Blumer, Herbert. 1951. “Collective Behavior.” *New Outline of the Principles of Sociology*, 166–222.

Bonilla, Tabitha, and Justin Grimmer. 2013. “Elevated Threat Levels and Decreased Expectations: How Democracy Handles Terrorist Threats.” *Poetics*, Topic Models and the Cultural Sciences, 41 (6): 650–69. doi:10.1016/j.poetic.2013.06.003.

Boudreau, Vince. 2005. “Precarious Regimes and Matchup Problems in the Explanation of Repressive Policy.” *Repression and Mobilization*, 33–57.

Boudreau, Vince. 2009. *Resisting Dictatorship: Repression and Protest in Southeast Asia*. Reissue edition. Cambridge, UK; New York: Cambridge University Press.

Bourdieu, Pierre. 1977. “Structures and the Habitus.” *Outline of a Theory of Practice*, 72–95.

Boykoff, Jules. 2007. “Limiting Dissent: The Mechanisms of State Repression in the USA.” *Social Movement Studies* 6 (3): 281–310.

Brockett, Charles D. 2005. *Political Movements and Violence in Central America*. Cambridge University Press. https://books.google.com/books?hl=en&lr=&id=gugQu8ujUMoC&oi=fnd&pg=PR1&dq=Brockett+CD.+1991.+The+structure+of+political+opportunities+and+peasant+mobilization+in+Central+America.+Comp.+Polit.&ots=eDgBVZxtIb&sig=nUD_5KhNXrxST-gJh1itF7Utarg.

Bromley, D. G., and A. D. Shupe. 1983. “Repression and the Decline of Social Movements: The Case of New Religions.” Social Movements of the Sixties and Seventies: 335–347.

Carey, S.C. 2006. “The Dynamic Relationship Between Protest and Repression.” *Political Research Quarterly* 59 (1): 1–11.

Carley, Michael. 1997. "Defining Forms of Successful State Repression of Social Movement Organizations: A Case Study of the FBI's COINTEL PRO and the American Indian Movement." *Research in Social Movements, Conflicts and Change* 20: 151–76.

Chang, Paul Y. 2008. "Unintended Consequences of Repression: Alliance Formation in South Korea's Democracy Movement (1970–1979)." *Social Forces* 87 (2): 651–77.

Christiansen, Jonathan. 2009. "Four Stages of Social Movements." *EBSCO Research Starters*.

Clawson, Dan. 2003. *The Next Upsurge: Labor and the New Social Movements*. Cornell University Press.

Collins, R. 2009. Violence: A Micro-sociological Theory. Princeton University Press.

Cunningham, David, and Barb Browning. 2004. "The Emergence of Worthy Targets: Official Frames and Deviance Narratives Within the FBI." In *Sociological Forum*, 19:347–69. Springer. http://link.springer.com/article/10.1023/B:SOFO.0000042553.21098.f6.

Curtis, Russell L., and Louis A. Zurcher. 1973. "Stable Resources of Protest Movements: The Multi-organizational Field." *Social Forces* 52 (1): 53–61.

Davenport, C. 1994. "Militarization, Political Conflict, and Political Development in the Third World." *Political Conflict, Political Development, and Public Policy, Westport, Corm*, 27–44.

Davenport, C. 1997. "From Ballots to Bullets: An Empirical Assessment of How National Elections Influence State Uses of Political Repression." *Electoral Studies* 16 (4): 517–540.

Davenport, C., S. A. Soule, and D. A. Armstrong. 2011. "Protesting While Black? The Differential Policing of American Activism, 1960 to 1990." American Sociological Review 76 (1): 152–178.

Davenport, Christian, and Molly Inman. 2012. "The State of State Repression Research Since the 1990s." *Terrorism and Political Violence* 24 (4): 619–34.

Davenport, Christian, Sarah A. Soule, and David A. Armstrong. 2011. "Protesting While Black? The Differential Policing of American Activism, 1960 to 1990." *American Sociological Review* 76 (1): 152–78.

Davenport, Christian. 2000. *Paths to State Repression: Human Rights Violations and Contentious Politics*. Rowman & Littlefield. https://books.google.com/books?hl=en&lr=&id=LsgKTh3S368C&oi=fnd&pg=PR7&dq=Davenport+C,+ed.+2000a.+Paths+to+State+Repression.+La&ots=Bpaymmw_6E&sig=SKsAKNhexbSp-XN1RVd7baNUlQo.

Davenport, Christian. 2007. "State Repression and Political Order." *Annu. Rev. Polit. Sci.* 10: 1–23.

Davenport, Christian. 2007. "State Repression and Political Order." *Annu. Rev. Polit. Sci.* 10: 1–23.

De Biasi, Rocco. 1998. "The Policing of Hooliganism in Italy." In *Policing Protest*. Univ. of Minnesota Press.

Del Corro, Luciano, and Rainer Gemulla. 2013. "ClausIE: Clause-based Open Information Extraction." In *Proceedings of the 22nd International Conference on World Wide Web*, 355–66. International World Wide Web Conferences Steering Committee. http://dl.acm.org/citation.cfm?id=2488420.

Della Porta D. 1996. Social movements and the state: thoughts on the policing of protest. In Comparative Perspectives on Social Movements, ed. D McAdam, JD McCarthy, MN Zald, pp. 62–92. Cambridge, UK: Cambridge Univ. Press

Della Porta, D. 1998. "Police Knowledge and Protest Policing: Some Reflections on the Italian Case." *Policing Protest. The Control of Mass Demonstrations in Western Democracies*: 228–252.

Della Porta, D. 1999. "Protest, Protesters, and Protest Policing: Public Discourses in Italy and Germany from the 1960s to the 1980s." How Social Movements Matter: 66–96.

Della Porta, D. 2006. Social Movements, Political Violence, and the State: A Comparative Analysis of Italy and Germany. Cambridge University Press.

Della Porta, D. and Fillieule. 2008. "Policing Social Protest." In Snow, D. A., S. A. Soule, and H. Kriesi. 2008. The Blackwell Companion to Social Movements. Wiley-Blackwell.

Della Porta, D., and H. Reiter. 1998. *Policing Protest: The Control of Mass Demonstrations in Western Democracies*. Vol. 6. Univ Of Minnesota Press.

Della Porta, Donatella, and Herbert Reiter Reiter. 1998. *Policing Protest: The Control of Mass Demonstrations in Western Democracies*. Vol. 6. U of Minnesota Press. https://books.google.com/books?hl=en&lr=&id=NMcGg_VCHlkC&oi=fnd&pg=PP7&dq=della+porta+1998&ots=pKuPGk2rNK&sig=XHlM8IWS8QdiMcjqPM6RohaGBgM.

Della Porta, Donatella, and Mario Diani. 2009. *Social Movements: An Introduction*. John Wiley & Sons. https://books.google.com/books?hl=en&lr=&id=Ig0gSeiKBvwC&oi=fnd&pg=PR5&dq=social+movements+della+porta+diani&ots=r7GDx2xwIr&sig=bXWSP99KloFwWV5ok9TCkM18GR4.

Della Porta, Donatella, and Massimiliano Andretta. 2001. "Movimenti Sociali e Rappresentanza: i Comitati Spontanei Dei Cittadini a Firenze." *Rassegna Italiana Di Sociologia* 42 (1): 41–76.

Della Porta, Donatella, and Sidney Tarrow. 2012. "Interactive Diffusion The Coevolution of Police and Protest Behavior With an Application to Transnational Contention." *Comparative Political Studies* 45 (1): 119–52. doi:10.1177/0010414011425665.

Della Porta, Donatella. 1995. "Social Movements and the State: Thoughts on the Policing of Protest." http://cadmus.eui.eu/handle/1814/1388.

Della Porta, Donatella. 2013. *Clandestine Political Violence*. Cambridge University Press.

Della, Porta D., and Mario Diani. 2006. "Social Movements: An Introduction." *Malden, MA: Blackwell*.

DiMaggio, Paul, Manish Nag, and David Blei. 2013. "Exploiting Affinities Between Topic Modeling and the Sociological Perspective on Culture: Application to Newspaper Coverage of U.S. Government Arts Funding." *Poetics*, Topic Models and the Cultural Sciences, 41 (6): 570–606. doi:10.1016/j.poetic.2013.08.004.

Duvall, Raymond, and Michael Stohl. 1983. "Governance by Terror." *The Politics of Terrorism*, 179–219.

Earl, J. 2011. "Political Repression: Iron Fists, Velvet Gloves, and Diffuse Control." Annual Review of Sociology 37: 261–284.

Earl, J., A. Martin, J.D. McCarthy, and S.A. Soule. 2004. “The Use of Newspaper Data in the Study of Collective Action.” *Annual Review of Sociology*: 65–80.

Earl, J., A. Martin, J.D. McCarthy, and S.A. Soule. 2004. “The Use of Newspaper Data in the Study of Collective Action.” *Annual Review of Sociology*, 65–80.

Earl, J., and S.A. Soule. 2006. “Seeing Blue: A Police-centered Explanation of Protest Policing.” *Mobilization: An International Quarterly* 11 (2): 145–164.

Earl, J., S.A. Soule, and J.D. McCarthy. 2003. “Protest Under Fire? Explaining the Policing of Protest.” *American Sociological Review*: 581–606.

Earl, Jennifer and Sarah A. Soule. 2006. “Seeing Blue: A Police-Centered Explanation of Protest Policing.” Mobilization 11(2): 145-164. Link

Earl, Jennifer, Sarah A. Soule, and John D. McCarthy. 2003. “Protest Under Fire? Explaining the Policing of Protest.” American Sociological Review 68(4): 581-606. Link

Earl, Jennifer. 2003. “Tanks, Tear Gas, and Taxes: Toward a Theory of Movement Repression.” *Sociological Theory* 21 (1): 44–68.

Eisinger, P. K. 1973. “The Conditions of Protest Behavior in American Cities.” The American Political Science Review: 11–28.

Feierabend, Ivo K., and Rosalind L. Feierabend. 1966. “Aggressive Behaviors Within Polities, 1948-1962: a Cross-national Study.” *Journal of Conflict Resolution*, 249–71.

Fernandez, L. (2008). Policing Dissent: Social Control and the Anti-globalization Movement. Brunswick, N.J: Rutgers University Press.

Fernandez, Luis. 2008. *Policing Dissent: Social Control and the Anti-Globalization Movement*. New Brunswick, N.J: Rutgers University Press.

Filleule, Olivier. 1997. Strategies de la rue: Les manifestations en France. Paris: Presses de Sciences Po.

Fillieule, Olivier, and Fabien Jobard. 1998. “The Policing of Protest in France: Toward a Model of Protest Policing.” *Policing Protest: The Control of Mass Demonstrations in Western Democracies*, 70–90.

Fillieule, Olivier. 1997. *Stratégies de La Rue: Les Manifestations En France*. Presses de Sciences po. http://www.cairn.info/strategies-de-la-rue--9782724607074.htm;978-2-72468-614-2;;Strat.

Flam, Helena. 1994. *States and Anti-Nuclear Movements*. illustrated edition edition. Edinburgh: Edinburgh University Press.

Franzosi, Roberto, Gianluca De Fazio, and Stefania Vicari. 2012. "Ways of Measuring Agency An Application of Quantitative Narrative Analysis to Lynchings in Georgia (1875–1930)." Sociological Methodology 42, no. 1: 1–42.

Franzosi, Roberto. 1987. "The Press as a Source of Socio-Historical Data: Issues in the Methodology of Data Collection from Newspapers." Historical Methods 20:5– 15.

Franzosi, Roberto. 1998. "Narrative Analysis-Or Why (And How) Sociologists Should Be Interested in Narrative." *Annual Review of Sociology* 24 (January): 517–54.

Funk, A. 1991. "„Innere Sicherheit ". Symbolische Politik Und Exekutive Praxis." Die Alte Bundesrepublik. Kontinuität Und Wandel, Opladen: 366–385.

Gamson, W. A. 1990 [1975]. The Strategy of Social Protest. Homewood, IL: Dorsey.

Gao, Jianbo, Kalev H. Leetaru, Jing Hu, Claudio Cioffi-Revilla, and Philip Schrodt. "Massive Media Event Data Analysis to Assess World-Wide Political Conflict and Instability." In Social Computing, Behavioral-Cultural Modeling and Prediction, edited by Ariel M. Greenberg, William G. Kennedy, and Nathan D. Bos, 284–92. Lecture Notes in Computer Science 7812. Springer Berlin Heidelberg, 2013. http://link.springer.com/chapter/10.1007/978-3-642-37210-0_31.

Geary, R. 1985. Policing Industrial Disputes: 1893 to 1985. Vol. 937. Cambridge University Press.

Gillham, P. F., and J. A. Noakes. 2007. "' More Than A March in a Circle': Transgressive Protests and the Limits of Negotiated Management." *Mobilization: An International Quarterly* 12 (4): 341–357.

Gitlin, Todd. 1980. The Whole World Is Watching: Mass Media in the Making and Unmaking of the New Left. Berkeley and Los Angeles: University of California Press.

Gold, Jim. 2015. “Mayors Deny Colluding on ‘Occupy’ Crackdowns.” *Msnbc.com*. Accessed July 3. http://www.nbcnews.com/id/45312298/ns/us_news-life/t/mayors-deny-colluding-occupy-crackdowns/.

Goldstein, R. J. 1978. Political Repression in Modern America from 1870 to the Present. GK Hall Boston.

Goodman, Amy. 2011. “From Tahrir Square to Occupy Wall Street.” *The Guardian*, October 26, sec. World news. http://www.theguardian.com/commentisfree/cifamerica/2011/oct/26/tahrir-square-occupy-wall-street.

Gould-Wartofsky, Michael. 2015. *The Occupiers: The Making of the 99 Percent Movement*. 1 edition. New York, NY: Oxford University Press.

Grimmer, Justin, and Brandon M. Stewart. 2013. “Text as Data: The Promise and Pitfalls of Automatic Content Analysis Methods for Political Texts.” *Political Analysis*, mps028.

Gurr, T. R. 1968. “A Causal Model of Civil Strife: A Comparative Analysis Using New Indices.” American Political Science Review 62:1104–24.

Hammond, Jesse, and Nils B. Weidmann. “Using Machine-coded Event Data for the Micro-level Study of Political Violence.” Research & Politics 1, no. 2 (July 1, 2014): 2053168014539924. doi:10.1177/2053168014539924.

Herman, Edward, and Noam Chomsky. 1988. Manufacturing Consent. New York: Pantheon.

Hopkins, Daniel J., and Gary King. 2010a. “A Method of Automated Nonparametric Content Analysis for Social Science.” American Journal of Political Science 54 (1): 229–47.

Huber, Evelyne, Charles Ragin, and John D. Stephens. 1993. “Social Democracy, Christian Democracy, Constitutional Structure, and the Welfare State.” *American Journal of Sociology*, 711–49.

Jackson, John Harold, and Cyril P. Morgan. 1978. *Organization Theory: a Macro Perspective for Management*. Prentice-Hall.

Jaime-Jimenez, Oscar, and Fernando Reinares. 1998. *The Policing of Social Protest in Spain: From Dictatorship to Democracy*. Minneapolis: University of Minnesota Press.

https://books.google.com/books?hl=en&lr=&id=NMcGg_VCHlkC&oi=fnd&pg=PA166&dq=Jaime-Jimenez+and+Reinares+(1998)&ots=pKuPGk3kRH&sig=KY22WmzTe8mzac_a37dMC_jN1wg.

Jenkins, J. Craig, and Charles Perrow. 1977. “Insurgency of the Powerless: Farm Worker Movements (1946–1972).” American Sociological Review 42:249– 68.

Jenkins, J. Craig, and Craig M. Eckert. 1986. “Channeling Black Insurgency: Elite Patronage and Professional Social Movement Organizations in the Development of the Black Movement.” American Sociological Review 51:812–29.

Jennifer Earl, and Sarah A. Soule. 2010. “The Impacts of Repression: The Effect of Police Presence and Action on Subsequent Protest Rates.” In *Research in Social Movements, Conflicts and Change*, 30:75–113. Research in Social Movements, Conflicts and Change 30. Emerald Group Publishing Limited. http://www.emeraldinsight.com/doi/abs/10.1108/S0163-786X%282010%290000030006.

Johnston, Hank. 2006. “‘ Let’s Get Small’: The Dynamics of (Small) Contention in Repressive States.” *Mobilization: An International Quarterly* 11 (2): 195–212.

Karstedt-Henke, S. 1980. “Theorien Zur Erklärung Terroristischer Bewegungen.” Politik Der Inneren Sicherheit: 198–234.

Keating, Joshua. 2015. “From Tahrir Square to Wall Street.” *Foreign Policy*. Accessed August 12. https://foreignpolicy.com/2011/10/05/from-tahrir-square-to-wall-street/.

Keertipati, Swetha, Bastin Tony Roy Savarimuthu, Maryam Purvis, and Martin Purvis. 2014. “Multi-level Analysis of Peace and Conflict Data in GDELT.” In Proceedings of the MLSDA 2014 2Nd Workshop on Machine Learning for Sensory Data Analysis, 33:33–33:40. MLSDA’14. New York, NY, USA: ACM, 2014. doi:10.1145/2689746.2689750.

King, Brayden G. and Sarah A. Soule. 2007. "Social Movements as Extra-Institutional Entrepreneurs: The Effect of Protest on Stock Price Returns." Administrative Science Quarterly 52: 413-42. Link

King, Brayden G., Keith G. Bentele and Sarah A. Soule. 2007. "Protest and Policymaking: Explaining Fluctuation in Congressional Attention to Rights Issues, 1960-1986." Social Forces 86(1):137-164.Link

King, Brayden G., Keith G. Bentele, and Sarah A. Soule. 2007. "Protest and Policymaking: Explaining Fluctuation in Congressional Attention to Rights Issues, 1960-1986." *Social Forces* 86 (1): 137–63.

King, Gary, Jennifer Pan, and Margaret Roberts. 2013. "How Censorship in China Allows Government Criticism but Silences Collective Expression." *American Political Science Review* 107 (2 (May)): 1–18.

Kitschelt, Herbert P. 1986. "Political Opportunity Structures and Political Protest: Anti-nuclear Movements in Four Democracies." *British Journal of Political Science* 16 (01): 57–85.

Koopmans, R. 2004. "Protest in Time and Space: The Evolution of Waves of Contention." *The Blackwell Companion to Social Movements*. In Snow, D. A., S. A. Soule, and H. Kriesi. 2008. The Blackwell Companion to Social Movements. Wiley-Blackwell.

Koopmans, Ruud. 1999. "Political. Opportunity. Structure. Some Splitting to Balance the Lumping." In *Sociological Forum*, 14:93–105. Springer. http://www.springerlink.com/index/N804356J76742311.pdf.

Kraska, Peter B., and Derek J. Paulsen. 1997. "Militarizing Mayberry and Beyond: Making Sense of American Paramilitary Policing." *Justice Quarterly* 14 (4): 607–29.

Kraska, Peter B., and Victor E. Kappeler. 1997. "Militarizing American Police: The Rise and Normalization of Paramilitary Units." *SOCIAL PROBLEMS-NEW YORK*-44: 1–18.

Kriesi, Hanspeter, Ruud Koopmans, Jan Willem Duyvendak, and Marco G. Giugni. 1995. New Social Movements in Western Europe: A Comparative Analysis. Minneapolis: University of Minnesota Press.

Kriesi, Hanspeter. 1995. "The Political Opportunity Structure of New Social Movements: Its Impact on Their Mobilization." *The Politics of Social Protest: Comparative Perspectives on States and Social Movements*, 167–98.

Kriesi, Hanspeter. 2009. “Charles Tilly: Contentious Performances, Campaigns and Social Movements” Swiss Political Science Review 15(2): 341–49.

Kwak, Haewoon, and Jisun An. 2014. “A First Look at Global News Coverage of Disasters by Using the GDELT Dataset.” In Social Informatics, edited by Luca Maria Aiello and Daniel McFarland, 300–308. Lecture Notes in Computer Science 8851. Springer International Publishing, http://link.springer.com/chapter/10.1007/978-3-319-13734-6_22.

Lacey, N., C. Wells and D. Meure. 1990. *Reconstructing Criminal Law.* Weidenfield and Nicholson. London.

Larson, Jeffrey and Sarah A. Soule. 2009. “Sector Level Dynamics and Collective Action in the United States: 1965-1975.” Mobilization 14(3): 293-314. Link

Laver, J. Garry. 2000. “Estimating Policy Positions from Political Texts.” *American Journal of Political Science*, 619–34. doi:10.2307/2669268.

Lee, Heeyoung, Angel Chang, Yves Peirsman, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky. 2013. “Deterministic Coreference Resolution Based on Entity-centric, Precision-ranked Rules.” *Computational Linguistics* 39 (4): 885–916.

Leetaru, Kalev, and Philip A. Schrodt. 2013. “GDELT: Global Data on Events, Location, and Tone, 1979–2012.” In Of: Paper Presented at the ISA Annual Convention, 2:4. http://eventdata.parusanalytics.com/presentations.dir/Schrodt.Leetaru.GDELT.Presentation.pdf.

Lieberson, Stanley , and Arnold R. Silverman. 1965. “The Precipitants and Underlying Conditions of Race Riots.” American Sociological Review 30:887–98.

Loveman, Mara. 1998. “High-Risk Collective Action: Defending Human Rights in Chile, Uruguay, and Argentina 1.” *American Journal of Sociology* 104 (2): 477–525.

Lowi, Theodore. 1971. “The Politics ofDisorder.” *NewYork: NortonCo. Luebbert, GM (1986) Comparative Democracy: Policymaking*.

Lundman, Richard J. 1980. *Police and Policing: An Introduction*. Holt, Rinehart and Winston New York. http://www.ncjrs.gov/App/abstractdb/AbstractDBDetails.aspx?id=62638.

Mahooney-Norris KA. 2000. Political repression: threat perception and transnational solidarity groups. See Davenport 2000, pp. 71–108

Manning, Christopher D., Surdeanu, Mihai, Bauer, John, Finkel, Jenny, Bethard, Steven J., and McClosky, David. 2014. "The Stanford CoreNLP Natural Language Processing Toolkit." In Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, pp. 55-60.

Marshall, Emily A. 2013. "Defining Population Problems: Using Topic Models for Cross-national Comparison of Disciplinary Development." *Poetics*, Topic Models and the Cultural Sciences, 41 (6): 701–24. doi:10.1016/j.poetic.2013.08.001.

Martin, Andrew W., John D. McCarthy, and Clark McPhail. 2009. "Why Targets Matter: Toward a More Inclusive Model of Collective Violence." American Sociological Review 74(5): 821-841. Link

Marx, Gary T. 1979. "External Efforts to Damage or Facilitate Social Movements: Some Patterns, Explanations, Outcomes, and Complications." *The Dynamics of Social Movements*, 94–125.

McAdam, Doug and Yang Su. 2002. "The War at Home: Antiwar Protests and Congressional Voting, 1965-1973." American Sociological Review 67: 696-721. Link

McAdam, Doug, Sidney Tarrow, and Charles Tilly. 2003. "Dynamics of Contention." *Social Movement Studies* 2 (1): 99–102.

McAdam, Doug. 1982. Political Process and the Development of Black Insurgency, 1930–1970. Chicago: University of Chicago Press.

McAdam, Doug. 2010. *Political Process and the Development of Black Insurgency, 1930-1970*. University of Chicago Press. https://books.google.com/books?hl=en&lr=&id=m_nkRogoE_sC&oi=fnd&pg=PR5&dq=political+process+and+black&ots=tP_tqmPmQ3&sig=YRNu9qusehifcbl6GlNBc1fG464.

McAdam, Doug. 2012. *Putting Social Movements in Their Place*. Cambridge University Press.

McCarthy, J. D., D. W. Britt, and M. Wolfson. 1991. "The Institutional Channeling of Social Movements by the State in the United States." *Research in Social Movements, Conflicts and Change* 13 (2).

McCarthy, John D. and Clark McPhail. 2006. "Places of Protest: The Public Forum in Principle and Practice" Mobilization 11(2): 229-47. Link

McCarthy, John D., and Clark McPhail. 1998. "The Institutionalization of Protest in the United States." *The Social Movement Society: Contentious Politics for a New Century*, 83–110.

McCarthy, John D., Clark McPhail, Jackie Smith, and Louis J. Crishock. 1999. "Electronic and Print Media Representations of Washington DC Demonstrations, 1982 and 1991: a Demography of Description Bias." *Acts of Dissent. New Developments in the Study of Protest, Maryland, Rowman/Littlefield*, 113–30.

McCarthy, John D., David W. Britt, and Mark Wolfson. 1991. "The Institutional Channeling of Social Movements by the State in the United States." *Research in Social Movements, Conflicts and Change* 13 (2).

McClintock, F., A. Normandeau, R. Philippe, J. Skolnick, and D. Szabo. 1974. "Police Et Violence Collective." Police, Culture Et Société. Montreal: Les Presses De l'Université De Montreal.

McPhail, C., D. Schweingruber, and J. McCarthy. 1998. "Policing Protest in the United States." *Policing Protest: The Control of Mass Demonstrations in Western Democracies*: 49–69.

McPhail, Clark, and John D. McCarthy. 2005a. "Protest Mobilization, Protest Repression, and Their Interaction." *Repression and Mobilization*, 3–32.

McPhail, Clark, David Schweingruber, and John McCarthy. 1998. "Policing Protest in the United States: 1960-1995." *Policing Protest: The Control of Mass Demonstrations in Western Democracies*, 49–69.

Meyer, John W., and Brian Rowan. 1977. "Institutionalized Organizations: Formal Structure as Myth and Ceremony." *American Journal of Sociology*, 340–63.

Miller, Ian Matthew. 2013. “Rebellion, Crime and Violence in Qing China, 1722–1911: A Topic Modeling Approach.” *Poetics*, Topic Models and the Cultural Sciences, 41 (6): 626–49. doi:10.1016/j.poetic.2013.06.005.

Minkoff, Debra C. 1995. *Organizing for Equality: The Evolution of Women’s and Racial-ethnic Organizations in America, 1955-1985*. Rutgers University Press.

Mohr, John W., Robin Wagner-Pacifici, Ronald L. Breiger, and Petko Bogdanov. 2013. “Graphing the Grammar of Motives in National Security Strategies: Cultural Interpretation, Automated Text Analysis and the Drama of Global Politics.” *Poetics*, Topic Models and the Cultural Sciences, 41 (6): 670–700. doi:10.1016/j.poetic.2013.08.003.

Molotch, Harvey, and Marilyn Lester. 1974. “News as Purposive Behavior: On the Strategic Use of Routine Events, Accidents, and Scandals.” American Sociological Review 39:101–12.

Molotch, Harvey, and others. 1979. “Media and Movements.” *The Dynamics of Social Movements*, 71–93.

Moore, W.H. 1998. “Repression and Dissent: Substitution, Context, and Timing.” *American Journal of Political Science*: 851–873.

Morris, Aldon D. 1986. *Origins of the Civil Rights Movements*. Simon and Schuster. https://books.google.com/books?hl=en&lr=&id=7vyHY9DWcu8C&oi=fnd&pg=PR5&dq=the+origins+of+the+civil+rights+movement+morris&ots=NQuDakFPhW&sig=e_Nm5D4QurbToN7yV5WyeQGZVms.

Myers, Daniel J., and Beth Schaefer Caniglia. 2004. “All the Rioting That’s Fit to Print: Selection Effects in National Newspaper Coverage of Civil Disorders, 1968-1969.” *American Sociological Review* 69 (4): 519–43. doi:10.1177/000312240406900403.

Nelson, Laura. 2014. “The Power of Place: Structure, Culture, and Continuities in U.S. Women’s Movements”. UC Berkeley, Ph.D. Dissertation.

Nepstad, Sharon, and Clifford Bob. 2006. “When Do Leaders Matter? Hypotheses on Leadership Dynamics in Social Movements.” *Mobilization: An International Quarterly* 11 (1): 1–22.

Noakes, J., and P.F. Gillham. 2007. “Police and Protester Innovation Since Seattle.” *Mobilization: An International Quarterly* 12 (4): 335–340.

Oberschall, Anthony. 1973. *Social Conflict and Social Movements*. Prentice hall.

Oliver, Pamela E., and Daniel J. Meyer. 1999. "How Events Enter the Public Sphere: Conflict, Location, and Sponsorship in Local Newspaper Coverage of Public Events." *American Journal of Sociology* 105 (1): 38–87. doi:10.1086/210267.

Oliver, Pamela. 2008. "Repression and Crime Control: Why Social Movement Scholars Should Pay Attention to Mass Incarceration as a Form of Repression." *Mobilization: An International Quarterly* 13 (1): 1–24.

Olzak, Susan and Sarah A. Soule. 2009. "Cross-Cutting Influences of Environmental Protest and Legislation." Social Forces 88(1):201-226. Link

Olzak, Susan. 1992. The Dynamics of Ethnic Competition and Conflict. Stanford, Calif.: Stanford University Press.

Ondetti, Gabriel. 2006. "Repression, Opportunity, and Protest: Explaining the Takeoff of Brazil's Landless Movement." *Latin American Politics and Society* 48 (2): 61–94.

Opp, K.D., and W. Roehl. 1990. "Repression, Micromobilization, and Political Protest." *Social Forces* 69 (2): 521–547.

Ortiz, David, Daniel Myers, Eugene Walls, and Maria-Elena Diaz. 2005. "Where Do We Stand with Newspaper Data?" *Mobilization: An International Quarterly* 10 (3): 397–419. doi:10.17813/maiq.10.3.8360r760k3277t42.

Osa, Maryjane, and Kurt Schock. 2007. "A Long, Hard Slog: Political Opportunities, Social Networks and the Mobilization of Dissent in Non-democracies." *Research in Social Movements, Conflicts and Change* 27: 123–53.

Parenti, Michael. 1986. The Politics of the Mass Media. New York: St. Martin's Press.

Piven, Frances Fox, and Richard Cloward. 1977. *Regulating the Poor: The Functions of Public Welfare*.

Pizzorno, A. 1978. Political Exchange and Collective Identity in Industrial Conflict. In Crouch and Pizzorno (eds.) *The Resurgence of Class Conflict in Western Europe.* New York: Homes & Meier, 277-98.

Pizzorno, A. 1981 Interests and Parties in Pluralism. In S. Berger (ed.), *Organizing Intersts in Western Europe.* Cambridge: Cambridge University Press, 3-46.

Quinn, Kevin M., Burt L. Monroe, Michael Colaresi, Michael H. Crespin, and Dragomir R. Radev. 2010. "How to Analyze Political Attention with Minimal Assumptions and Costs." American Journal of Political Science 54 (1): 209–28.

Rasler, Karen. 1996. "Concessions, Repression, and Political Protest in the Iranian Revolution." *American Sociological Review* 61 (1) (February 1): 132–152. doi:10.2307/2096410.

Rasler, Karen. 1996. "Concessions, Repression, and Political Protest in the Iranian Revolution." *American Sociological Review*, 132–52.

Reiner, Robert. 1998. "Policing, Protest, and Disorder in Britain." *Policing Protest: The Control of Mass Demonstrations in Western Democracies*, 35–48.

Ron, James. 2000. "Savage Restraint: Israel, Palestine and the Dialectics of Legal Repression." *SOCIAL PROBLEMS-NEW YORK-* 47 (4): 445–72.

Rubinstein, Jonathan. 1980. *City Police*. Macmillan. https://books.google.com/books?hl=en&lr=&id=fA9rETs_wuIC&oi=fnd&pg=PR9&dq=police+Rubinstein&ots=pBfIjP800K&sig=xrXVRw889ky6qXTqsgYdZYIjKH0.

Schrodt, P A, and J Yonamine. 2013. "Automated Coding of Very Large Scale Political Event Data." Proceedings of New Directions in Analyzing Text as Data conference. London.

Segura-Diaz, Jennifer. 2015. "Minority Participation and Political Participation: An Analysis of the Emergence of Occupy the Hood". UC Berkeley, Honor's Thesis.

Shorter, Edward, and Charles Tilly. 1974. Strikes in France. Cambridge: Cambridge University Press.

Shultziner, Doron. 2014. "A Multi-Stage Approach to Social Movements." *Mobilizing Ideas*. https://mobilizingideas.wordpress.com/2014/08/04/a-multi-stage-approach-to-social-movements/.

Skocpol, Theda. 1992. "Protecting Mothers and Soldiers: The Political Origins of Social Policy in the United States." *Protecting Mothers and Soldiers. The Political Origins of Social Policy in the United States*.

Skolnick, Jerome H. 1966. "Social Research on Legality - A Reply to Auerbach." *Law & Society Review* 1: 105.

Snow, D. A., S. A. Soule, and H. Kriesi. 2008. The Blackwell Companion to Social Movements. Wiley-Blackwell.

Soule, S.A., and C. Davenport. 2009. "Velvet Glove, Iron Fist, or Even Hand? Protest Policing in the United States, 1960-1990." *Mobilization: An International Quarterly* 14 (1): 1–22.

Soule, Sarah A. 2009. Contention and Corporate Social Responsibility. New York, NY: Cambridge University Press. Link

Soule, Sarah A. and Brayden G King. 2008. "Competition and Resource Partitioning in Three Social Movement Industries." American Journal of Sociology 113(6): 1568-1610. Link

Soule, Sarah A. and Christian Davenport. 2009. "Velvet Glove, Iron Fist, or Even Hand? Protest Policing in the United States, 1960-1990." Mobilization 14(1): 1-22. Link

Soule, Sarah A. and Jennifer Earl. 2005. "A Movement Society Evaluated: Collective Protest in the United States, 1960-1986." Mobilization 10(3): 345-364. Link

Sparck Jones, Karen. 1972. "A Statistical Interpretation of Term Specificity and Its Application in Retrieval." *Journal of Documentation* 28 (1): 11–21.

Spilerman, Seymour. 1970. "The Causes of Racial Disturbances: A Comparison of Alternative Explanations." American Sociological Review 35:627–49.

Spilerman, Seymour. 1976. "Structural Characteristics of Cities and the Severity of Racial Disorders." American Sociological Review 41:771–93.

Starr, Amory, and Luis Fernandez. 2009. "Legal Control and Resistance Post-Seattle." *Social Justice* 36 (1 (115)): 41–60.

Starr, Amory, Luis A. Fernandez, and Christian Scholl. 2011. *Shutting down the Streets: Political Violence and Social Control in the Global Era*. NYU Press. https://books.google.com/books?hl=en&lr=&id=hYqgIYstSwEC&oi=fnd&pg=PA1&dq=starr+fernandez+scholl&ots=yxmHQwDeNT&sig=p_HxrKNgWRZ7lCIhLxURFmno8IE.

Stockdill BC. 1996. *Multiple oppressions and their influence on collective action: the case of the AIDS movement*. PhD thesis. Northwest. Univ., Evanston, IL

Tarrow, Sidney. 2013. “Contentious Politics.” In *The Wiley-Blackwell Encyclopedia of Social and Political Movements*. Blackwell Publishing Ltd. http://onlinelibrary.wiley.com/doi/10.1002/9780470674871.wbespm051/abstract.

Tauber, Ronald K. 1967. “Danger and the Police: A Theoretical Analysis.” *Issues in Criminology*, 69–81.

Tilly, C. 1978. From Mobilization to Revolution. McGraw-Hill New York.

Tilly, Charles. 2005. “Social Movements, 1768-2004.” http://www.citeulike.org/group/2546/article/781106.

Tilly, Charles. 2006. *Contentious Politics*. 1 edition. New York, NY: Oxford University Press.

Tilly, Charles. 2008. *Contentious Performances*. Cambridge University Press Cambridge. http://www.langtoninfo.com/web_content/9780521515849_frontmatter.pdf.

Tilly, Charles. 2010. *Regimes and Repertoires*. University of Chicago Press. https://books.google.com/books?hl=en&lr=&id=LUTsPzvWMAwC&oi=fnd&pg=PR5&dq=regimes+and+repertoires&ots=tNRHU-XbUj&sig=YC86Pb7G4SwdLzRStFpfbMpV4v8.

United States Department of Justice. Office of Justice Programs. Bureau of Justice Statistics. Law Enforcement Management and Administrative Statistics (LEMAS), 2007. ICPSR31161-v1. Ann Arbor, MI: Inter-university Consortium for Political and Social Research [distributor], 2011-07-07. http://doi.org/10.3886/ICPSR31161.v1

Van Dyke, Nella, Sarah A. Soule, and Verta A. Taylor. 2004. “The Targets of Social Movements: Beyond a Focus on the State.” Research in Social Movements, Conflicts, and Change 25: 27-51. Link

Vavliakis, Konstantinos N., Andreas L. Symeonidis, and Pericles A. Mitkas. 2013. “Event Identification in Web Social Media through Named Entity Recognition and Topic Modeling.” *Data & Knowledge Engineering* 88 (November): 1–24. doi:10.1016/j.datak.2013.08.006.

Vitale, A. S. 2007. “The Command and Control and Miami Models at the 2004 Republican National Convention: New Forms of Policing Protests.” Mobilization: An International Quarterly 12 (4): 403–415.

Waddington, P. A. J. 1994. “Coercion and Accommodation: Policing Public Order After the Public Order Act.” *The British Journal of Sociology* 45 (3) (September 1): 367–385. doi:10.2307/591654.

Waddington, P. A. J. 1999. “Swatting Police Paramilitarism: A Comment on Kraska and Paulsen.” http://www.tandfonline.com/doi/abs/10.1080/10439463.1999.9964808.

Waddington, P.A.J. 1998. “Controlling Protest in Contemporary and Comparative Perspective.” In Della Porta, D., and H. Reiter (eds.). 1998. *Policing Protest: The Control of Mass Demonstrations in Western Democracies*. Vol. 6. Univ Of Minnesota Press.

Walker, Edward T., Andrew W. Martin, and John D. McCarthy. 2008. “Confronting the State, the Corporation, and the Academy: The Influence of Institutional Targets on Social Movement Repertoires.” American Journal of Sociology 114(1): 35:76. Link

Wilson, J. Q. 1978. Varieties of Police Behavior: The Management of Law and Order in Eight Communities. Harvard University Press.

Winter, Martin. 1998. “Police Philosophy and Protest Policing in the Federal Republic of Germany, 1960-1990.” In Della Porta, D., and H. Reiter (eds.). 1998. *Policing Protest: The Control of Mass Demonstrations in Western Democracies*. Vol. 6. Univ Of Minnesota Press.

Wisler, D. and H. Kriesi. 1998. “Public Order, Protest Cycles, and Political Process: Two Swiss Cities Compared.” Della Porta, D., and H. Reiter. 1998. *Policing Protest: The Control of Mass Demonstrations in Western Democracies*. Vol. 6. Univ Of Minnesota Press.

Worden, R. E. 1989. “Situational and Attitudinal Explanations of Police Behavior: A Theoretical Reappraisal and Empirical Assessment.” Law and Society Review: 667–711.

Zirn, Cäcilia, and Heiner Stuckenschmidt. 2014. “Multidimensional Topic Analysis in Political Texts.” *Data & Knowledge Engineering*, Special Issue on Natural

Language Processing and Information Systems (NLDB 2012), 90 (March): 38–53. doi:10.1016/j.datak.2013.07.003.

Zurcher, Louis A., and Russell L. Curtis. 1973. “A Comparative Analysis of Propositions Describing Social Movement Organizations*.” *Sociological Quarterly* 14 (2): 175–88. doi:10.1111/j.1533-8525.1973.tb00852.x.

Zwerman, Gilda, and Patricia Steinhoff. 2005. “When Activists Ask for Trouble: State-dissident Interactions and the New Left Cycle of Resistance in the United States and Japan.” *Repression and Mobilization*, 85–107.

Zwerman, Gilda. 1989. “Domestic Counterterrorism: US Government Responses to Political Violence on the Left in the Reagan Era.” *Social Justice*, 31–63.

Methodological Appendix A

Table of Contents:

Description and code for coreference resolution

Description and code for SVO extraction

Description and code for actor and verb replacements dictionaries

Description and code for Structural Topic Modeling of Contentious Performances

Description and code for Structural Topic Modeling of Contentious Performances

City/Town	State	Campaign Start	Protester TUAs	Police TUAs	Population	Obama_Vote2008	Ob ɔter 5-cale)	Next_Election	Prior_Election	Political Instability Score	Govt_Type	Centers of Power Score	Police Budget per1,000 Residents	Police Budget Score	Officers per 1,000 Residents	offic ers	Community Policing Index	Com Pol Score	Violent Crimes per 1,000 Residents	Violent Crime Score	Proportion of Officers Non-White	Non white Cop Score
Albany	NY	10/21/11	60	45	97856	63.2	1	2013-11	11/3/09	1	MC	2	447253.29	3	3.45	3	0	1	9.6	3	0.1	1
Albuquerque	NM	10/1/11	1		545852	60	3	2013-10	10/6/09	1	MC	1	274669.69	2	1.76	1	10	3	7.71	2	0.42	3
Allentown	PA	10/3/11	5	5	118032	57	1	2013-11	11/3/09	1	MC	1	217199.11	1	1.75	1	11	3	5.48	2	0.07	1
Anaheim	CA	10/7/11	2		336265	48	2	2014-11	11/2/10	1	CM	2	340150.99	3	1.19	1	7	2	3.81	1	0.3	3
Anchorage	AK	10/23/11	7	4	291826	37	1	2012-05	4/5/11	2	MC	1	259469.44	2	1.37	1	6	2	8.18	3	0.15	2
Ann Arbor	MI	10/6/11	1		113934	69.8	1	2012-11	11/2/10	2	MC	1	114101.15	1	1.23	1	6	2	2.29	1	0.21	2
Arcata	CA	11/1/11	17	6	17231	62.8	1	2011-12	11/1/08	2	CO	3	211833.09	1	5.17	3	6	2	2.79	1	0.12	1
Asheville	NC	9/28/11	23	9	83393	56.7	1	2013-11	11/3/09	1	CO	3	214049.39	1	2.46	2	0	1	5.32	2	0.1	1
Ashtabula	OH	11/7/11	4		19124	54.2	1	2011-11	11/6/07	2	CM	2	15948.55	1	1.67	1	0	1	4.5	2	0	1
Atlanta	GA	10/7/11	176	95	420003	67	2	2013-11	12/1/09	1	MC	2	440219.83	3	4.09	3	10	3	14.52	3	0.59	3
Augusta	ME	10/15/11	8	5	19136	56.4	1	2011-11	6/16/11	3	MC	1	189354.31	1	2.14	2	6	2	3.61	1	0.02	1
Aurora	CO	10/22/11	3		325078	41	1	2011-11	8/1/11	3	CM	2	231857.38	2	1.93	2	10	3	4.45	1	0.14	1
Austin	TX	10/6/11	70	67	790390	64	4	2012-05	5/9/09	2	CM	3	249259.97	2	1.82	2	12	3	4.39	1	0.31	3
Bakersfield	CA	10/15/11	7		347483	40	1	2012-06	11/2/10	2	CM	2	200807.39	1	1.07	1	8	2	5.37	2	0.21	2
Baltimore	MD	10/4/11	25	16	620961	57	3	2011-11	9/13/11	3	MC	1	558477.7	3	3.08	3	11	3	14.31	3	0.81	3
Bangor	ME	10/29/11	24	5	33039	52.3	1	2011-11	10/15/10	3	CO	3	217684.74	1	2.39	2	3	1	2.09	1	0	1
Baton Rouge	LA	10/22/11	6		229493	50.5	1	2012-11	10/4/08	1	MC	2	NA	NA	2.79	3	8	2	10.75	3	0	1

Bellingham	WA	10/14/11	3	3	80885	58	1	2011-11	9/16/11	3	MC	1	247448.27	2	1.37	1	7	2	2.46	1	0.07	1
Bend	OR	10/15/11	7	1	76639	49	1	2012-11	1/5/11	2	CO	3	233562.55	2	1.1	1	6	2	2.66	1	0.02	1
Bethlehem	PA	10/21/11	6	4	74982	57	1	2013-11	11/3/09	1	MC	1	129674.38	1	2.01	2	9	2	2.92	1	0.06	1
Binghamton	NY	10/13/11	8	1	47376	52.7	1	2013-11	11/3/09	1	MC	1	213544.69	1	3.06	3	0	1	5.66	2	0.03	1
Birmingham	AL	10/15/11	5	2	212237	52	1	2013-08	8/23/11	2	MC	1	390302.39	3	4.31	3	10	3	14.9	3	0.53	3
Bloomington	IL	10/9/11	23	1	80405	49.7	1	2013-11	11/3/09	1	CM	2	192263.49	1	1.31	1	6	2	4.49	1	NA	NA
Boise	ID	11/5/11	12	3	205671	45.9	1	2011-11	11/6/07	2	MC	2	204209.64	1	1.39	1	5	2	2.48	1	0.06	1
Boston	MA	9/30/11	67	65	617594	77	4	2013-11	11/3/09	1	MC	2	438303.08	3	3.53	3	9	2	8.5	3	0.35	3
Brattleboro	VT	10/7/11	4	1	12046	73.4	1	2012-03	3/3/09	2	CO	3	288320.71	2	3.64	3	0	1	6.61	2	0	1
Buffalo	NY	10/8/11	11	4	261310	55	1	2013-11	11/3/09	1	MC	1	240350.93	2	2.83	3	7	2	12.44	3	0.31	3
Burlington	VT	10/28/11	13	16	42417	71.8	1	2012-03	3/3/09	2	MC	1	NA	NA	2.36	2	0	1	2.17	1	NA	NA
Canton	OH	10/15/11	1		73007	50.2	1	2011-11	11/6/07	2	MC	1	235231.02	2	2.51	2	0	1	11.09	3	0.15	2
Cedar Falls	IA	10/15/11	4		39260	60.5	1	2011-11	11/4/09	2	MC	1	NA	NA	1.07	1	0	1	1.81	1	NA	NA
Cedar Rapids	IA	10/15/11	7	2	126326	60	1	2013-11	11/3/09	1	CM	2	194839.05	1	1.59	1	2	1	2.83	1	0.03	1
Chapel Hill	NC	10/15/11	12	9	57233	72.1	1	2011-11	11/3/09	2	CM	2	203746.46	1	2.06	2	0	1	1.85	1	0.18	2
Charleston	SC	10/19/11	23	22	120083	53.5	1	2011-11	11/6/07	2	MC	1	308120.22	3	3.17	3	0	1	3.31	1	0.25	2
Charleston	WV	10/15/11	12		51400	49.2	1	2015-05	5/17/11	1	MC	2	241597.96	2	3.56	3	0	1	9.92	3	0.1	1
Charlotte	NC	9/17/11	50	22	731424	62	4	2011-11	11/3/09	2	CM	2	238222.66	2	2.29	2	12	3	6.54	2	0.18	2
Chattanooga	TN	10/7/11	8	2	167674	43.5	1	2013-03	3/3/09	2	MC	1	251894.25	2	2.53	2	0	1	8.71	3	0.24	2
Chicago	IL	9/23/11	160	18	2695598	76	5	2015-02	2/22/11	1	MC	1	451625.8	3	4.95	3	10	3	11.33	3	0.45	3
Chico	CA	10/9/11	2		86187	49.7	1	2014-11	11/2/10	1	CO	3	264603	2	1.18	1	5	2	2.84	1	0.14	1

Cincinnati	OH	10/8/11	55	37	296943	53	1	2013-11	11/3/09	1	CM	2	464954.08	3	3.62	3	8	2	10.33	3	0.31	3
Claremont	CA	11/22/11	1	1	34926	68.7	1	2013-03	3/1/11	2	CO	3	253552.28	2	1.26	1	11	3	0.92	1	0.18	2
Clarksville	TN	10/15/11	2		132929	46	1	2014-11	11/2/10	1	MC	1	185074.39	1	1.86	2	1	1	6.64	2	0.2	2
Cleveland	OH	10/6/11	24	11	396815	69	2	2013-11	11/3/09	1	MC	1	456167.23	3	4.07	3	8	2	13.67	3	0.34	3
Coachella Valley	CA	10/24/11	6	15	40704	50.8	1	2012-11	11/2/10	2	CO	3	NA	NA	0.79	1	0	1	6.78	2	NA	NA
Colombus	OH	10/13/11	9	2	787033	60	4	2011-11	11/6/07	2	CM	3	304365.08	2	2.4	2	7	2	6.59	2	0.14	1
Colorado Springs	CO	9/30/11	16	41	416427	40	2	2015-04	5/17/11	1	CM	2	177702.21	1	1.6	1	8	2	4.48	1	0.18	2
Columbia	MO	9/24/11	3		108500	55.2	1	2013-04	4/6/10	2	CM	2	158375.88	1	1.37	1	12	3	5.36	2	0.11	1
Columbia	SC	10/15/11	44	34	129272	72	1	2014-04	4/20/10	2	CM	3	190309.99	1	2.76	3	8	2	5.92	2	0.04	1
Corpus Christi	TX	10/21/11	4	2	NA	47.4	NA	2012-11	5/16/11	2	CM	2	206261.23	1	1.44	1	0	1	6.58	2	NA	NA
Dallas	TX	10/6/11	158	72	1197816	57	4	2013-06	6/18/11	2	CM	2	302971.25	2	2.83	3	10	3	6.95	2	0.39	3
Dayton	OH	10/5/11	13		141527	51.8	1	2013-11	11/3/09	1	CM	2	355717.28	3	3.5	3	0	1	9.57	3	0.08	1
Denver	CO	9/23/11	122	174	600158	74	3	2015-06	6/7/11	1	MC	2	291003.87	2	2.54	3	9	2	6.18	2	0.32	3
Des Moines	IA	10/9/11	34	4	203433	56	1	2011-11	11/6/07	2	CM	3	229468.45	2	1.87	2	12	3	5.25	2	0.09	1
Detroit	MI	10/14/11	29		713777	45	3	2013-08	11/3/09	1	MC	1	571037.41	3	3.15	3	11	3	21.36	3	0.94	3
Duluth	MN	10/16/11	10	15	86265	65	1	2011-11	4/6/07	1	MC	1	186736.22	1	2.05	2	9	2	4.14	1	0.07	1
Easton	PA	10/15/11	5		26800	55.5	1	2011-11	11/6/07	2	MC	1	NA	NA	2.31	2	0	1	3.58	1	NA	NA
El Paso	TX	10/17/11	15	15	649121	66	3	2013-05	5/9/09	1	CM	2	179185.98	1	2.08	2	8	2	4.4	1	0.65	3
Eugene	OR	10/15/11	59	36	156185	62.9	1	2012-11	11/4/08	1	CM	2	270051.18	2	1.21	1	0	1	2.95	1	0.14	1
Eureka	CA	10/13/11	10	42	27191	62.8	1	2014-11	11/2/10	1	MC	1	309457.14	3	1.8	2	1	1	4.49	1	0.02	1
Everett	WA	10/29/11	6	1	103019	58	1	2013-11	11/3/09	1	MC	1	241833.33	2	1.9	2	6	2	4.37	1	0.1	1

Fairbanks	AK	10/18/11	2	1	31535	30	1	2013-10	10/12/10	2	MC	1	189082.67	1	1.49	1	0	1	5.61	2	NA	NA
Fayetteville	AR	10/15/11	2		73580	42.4	1	2012-11	11/25/08	1	MC	1	170919.14	1	1.64	1	5	2	4.06	1	0.05	1
Flint	MI	10/14/11	4		102434	65	1	2011-11	8/4/09	2	MC	1	312396.27	3	2.56	3	10	3	23.35	3	0.48	3
Fort Collins	CO	10/10/11	4	1	143986	54	1	2013-04	4/5/11	2	CM	2	195306.99	1	1.13	1	9	2	2.92	1	0.1	1
Fort Lauderdale	FL	10/8/11	9		165521	62.7	1	2012-01	2/10/09	2	CM	2	534776.19	3	3.08	3	10	3	9.45	3	0.22	2
Fort Myers	FL	10/8/11	5	3	62298	44.4	1	2013-11	9/15/09	1	CM	2	400703.07	3	3.26	3	4	1	12.36	3	0.19	2
Fort Wayne	IN	10/15/11	9	1	253691	47	1	2011-11	11/6/07	2	MC	1	192411.39	1	1.81	2	4	1	3.09	1	0.18	2
Fort Worth	TX	10/15/11	4	16	741206	44	3	2012-11	6/18/11	2	CM	2	260555.55	2	2	2	9	2	6.16	2	0.28	2
Frederick	MD	11/17/11	3		65239	49	1	2013-11	11/3/09	1	MC	1	342347.94	3	2.22	2	8	2	4.69	2	0.16	2
Fresno	CA	10/6/11	8	11	494665	50	2	2012-06	11/4/08	1	CM	2	201075.88	1	1.81	2	10	3	5.89	2	0.39	3
Gainesville	FL	10/12/11	10	3	124354	60	1	2012-04	4/13/10	2	CM	2	NA	NA	2.36	2	9	2	7.33	2	NA	NA
Grand Rapids	MI	10/8/11	26	3	188040	49.4	1	2011-11	8/7/07	1	CM	2	241105.33	2	1.78	1	9	2	7.42	2	0.14	1
Greeley	CO	10/24/11	2		92889	44.6	1	2011-11	11/20/07	2	CM	2	228230.51	2	1.54	1	3	1	4.7	2	0.13	1
Greensboro	NC	10/15/11	12		269666	59	2	2013-10	11/3/09	1	CM	2	221486.98	2	2.2	2	8	2	0.01	1	0.24	2
Harrisburg	PA	10/15/11	23	10	49528	83.1	1	2013-11	11/3/09	1	MC	2	NA	NA	NA	NA	0	1	14.07	3	NA	NA
Hartford	CT	10/7/11	14		124775	92	1	2011-11	11/6/07	2	MC	2	298423.29	2	3.34	3	13	3	13.14	3	0.37	3
Honolulu	HI	11/5/11	29	16	337256	70	2	2012-09	7/20/10	2	MC	2	561484.32	3	5.73	3	9	2	4.05	1	0.82	3
Houston	TX	10/10/11	37	11	2099451	50	5	2011-11	12/12/09	2	MC	1	286679.21	2	2.41	2	10	3	9.95	3	0.46	3
Huntington	WV	10/7/11	3	1	49138	39.8	1	2012-11	11/2/10	2	MC	1	225273.47	2	2.18	2	0	1	6.63	2	NA	NA

Indianapolis	IN	10/8/11	13	7	820445	64	4	2011-11	11/6/07	2	MC	2	250040.65	2	1.93	2	10	3	11.18	3	0.16	2
Iowa City	IA	10/7/11	14	4	67862	70.2	1	2011-11	1/2/08	2	CO	3	142471.46	1	1.08	1	5	2	2.4	1	0.05	1
Ithaca	NY	11/22/11	1		30014	69.4	1	2011-11	11/6/07	2	MC	1	366827.81	3	2.1	2	0	1	2.3	1	NA	NA
Jackson	MS	10/15/11	11	3	173514	50	1	2013-06	6/2/09	1	MC	2	321352.74	3	2.59	3	9	2	9.34	3	0.69	3
Jacksonville	FL	11/5/11	41	5	864263	49	3	2015-05	5/17/11	1	MC	1	344541.74	3	1.92	2	5	2	6	2	0.24	2
Jersey City	NJ	10/11/11	16	1	247597	72.7	2	2011-11	5/12/09	2	CO	3	384893.12	3	NA	NA	0	1	7.7	2	NA	NA
Johnson City	TN	10/15/11	2		63152	30.2	1	2013-04	4/2/09	1	CM	2	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
Kalamazoo	MI	10/12/11	11		74262	59	1	2013-11	11/3/09	1	CM	2	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
Kansas City	MO	10/3/11	12	2	459787	49	2	2015-03	3/22/11	1	CM	2	93521.57	1	0.82	1	13	3	12.04	3	0.59	3
Lansing	MI	10/10/11	13	2	114297	66	1	2014-08	11/3/09	1	MC	2	292642.74	2	2.19	2	10	3	10.23	3	0.22	2
Las Cruces	NM	10/15/11	6	13	97618	58	1	2011-11	11/6/07	2	CM	2	29051.6	1	1.81	2	10	3	4.34	1	0.62	3
Las Vegas	NV	10/6/11	41	1	583756	58	3	2015-06	6/7/11	1	CM	2	884435.5	3	5.04	3	8	2	18.52	3	0.18	2
Lawrence	KS	10/8/11	5	10	87643	64.2	1	2013-04	4/7/09	1	CM	2	165318.05	1	1.57	1	14	3	3.82	1	NA	NA
Lexington	KY	9/29/11	7		295803	52	1	2012-11	11/2/10	2	MC	1	167679.16	1	1.93	2	8	2	4.59	2	0.11	1
Lincoln	NE	10/15/11	3	4	258379	52	1	2015-05	5/3/11	1	MC	2	116108.51	1	1.23	1	7	2	3.74	1	0.06	1
Little Rock	AR	10/15/11	27	20	193524	55	1	2014-11	11/10/10	1	CM	3	237697.88	2	2.81	3	7	2	15.01	3	0.29	3
Long Beach	CA	10/15/11	21	36	462257	69	3	2014-04	4/13/10	2	CM	2	389823.09	3	2.12	2	12	3	6.18	2	0.43	3
Los Angeles	CA	10/1/11	258	223	3792621	69	5	2013-03	3/3/09	2	MC	1	323591.06	3	2.56	3	10	3	5.29	2	0.59	3
Louisville	KY	10/4/11	12	3	597337	56	3	2014-11	11/2/10	1	MC	1	251655.59	2	1.98	2	10	3	6.84	2	0.16	2
Lubbock	TX	10/15/11	3	7	236056	31.2	1	2012-05	5/8/10	2	CM	2	35712.67	1	NA	NA	0	1	8.31	3	NA	NA

Madison	WI	10/7/11	11	2	233209	73	2	2015-04	4/5/11	1	MC	2	214856.42	1	1.72	1	0	1	3.49	1	0.19	A 2
Memphis	TN	10/15/11	16	5	646889	63	3	2011-10	10/15/09	2	MC	1	308821.74	3	2.39	2	8	2	15.98	3	0.7	3
Merced	CA	10/15/11	2	2	255793	52.9	1	2011-11	11/3/09	2	CM	2	168104.68	1	0.45	1	5	2	1.97	1	0.11	1
Miami	FL	10/14/11	51		399457	58	2	2013-11	11/3/09	1	MC	1	45689.4	1	7.74	3	11	3	12.14	3	0.29	3
Milwaukee	WI	10/15/11	28	10	594833	67	3	2012-04	4/1/08	1	MC	1	356368.61	3	3.34	3	8	2	10.03	3	0.34	3
Minneapolis	MN	10/7/11	104	38	382578	63	2	2013-11	11/3/09	1	MC	2	312844.37	3	2.23	2	12	3	10.13	3	0.18	2
Missoula	MT	10/8/11	23	4	66788	61.9	1	2013-11	11/3/09	1	CM	2	166104.03	1	1.5	1	0	1	2.87	1	NA	NA
Mobile	AL	11/5/11	6	9	195111	45.3	1	2013-08	8/25/09	1	CM	2	213759.25	1	2.73	3	0	1	8.3	3	0.29	3
Mosier	OR	11/5/11	1		433	52.2	1	2012-11	11/2/10	2	CO	3	NA	NA	36.95	3	0	1	0	1	NA	NA
Muncie	IN	10/19/11	4	1	70085	57	1	2011-11	11/6/07	2	MC	1	140816.45	1	1.6	1	8	2	7.32	2	0.05	1
Murfreesboro	TN	10/20/11	2		108755	39.8	1	2014-04	4/6/10	2	CO	3	166318.74	1	1.72	1	0	1	6.01	2	0.16	2
Muskegon	MI	10/17/11	5		38401	63.9	1	2013-11	11/3/09	1	CM	2	228587.51	2	2.37	2	0	1	9.48	3	NA	NA
Nashville	TN	10/8/11	62	66	601222	60	3	2015-08	8/4/11	2	MC	2	270125.84	2	2.18	2	12	3	12.04	3	0.16	2
Newark	NJ	11/18/11	12	7	277140	76	2	2014-05	5/11/10	2	MC	1	469077	3	4.43	3	8	2	11.7	3	0.73	3
New Haven	CT	10/15/11	17	8	129779	88	1	2011-11	11/3/09	2	MC	1	268102.26	2	NA	NA	0	1	13.47	3	NA	NA
New Orleans	LA	10/6/11	38	36	343829	79	2	2014-02	12/11/09	1	MC	1	325743.32	3	4.65	3	8	2	7.99	2	0.53	3
New Paltz	NY	10/15/11	1		6818	60.6	1	2015-05	5/3/11	1	CM	2	333318.42	3	4.69	3	0	1	4.84	2	0.03	1
New York City	NY	9/17/11	256	85	8175133	77	5	2013-11	11/3/09	1	MC	1	464824.24	3	4.41	3	11	3	6.26	2	0.44	3
Norfolk	VA	10/6/11	8	9	242803	71	2	2014-05	5/4/10	2	CM	2	200397.03	1	3.16	3	6	2	5.86	2	0.24	2

Norman	OK	12/3/11	2	1	110925	38	1	2013-03	3/2/10	2	CM	2	146014.39	1	1.22	1	0	1	1.72	1	0.06	1
Northampton	MA	10/6/11	1	2	28549	71.8	1	2011-11	11/3/09	2	MC	1	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
Oakland	CA	10/10/11	463	345	390724	79	2	2014-11	11/2/10	1	MC	1	491395.46	3	2.06	2	6	2	17.02	3	0.5	3
Ogden	UT	11/5/11	3		83796	34.8	1	2011-11	11/6/07	2	MC	1	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
Oklahoma City	OK	10/10/11	31	18	579999	42	2	2014-03	3/2/10	1	CM	3	226695.9	2	1.78	1	10	3	8.81	3	0.14	1
Olympia	WA	10/15/11	10	7	252264	60	1	2011-11	11/6/07	2	MC	2	108001.01	1	0.38	1	0	1	0.5	1	0.03	1
Omaha	NE	10/15/11	14	20	408958	52	2	2013-05	5/12/09	1	MC	1	219768.07	2	1.87	2	10	3	5.65	2	0.18	2
Orlando	FL	10/15/11	18	16	238300	59	1	2012-04	1/29/08	1	MC	1	459044.48	3	3.07	3	7	2	10.87	3	0.38	3
Palo Alto	CA	10/13/11	2		64403	69.6	1	2012-11	1/4/11	2	CM	2	434460.01	3	1.44	1	4	1	0.99	1	0.28	2
Pensacola	FL	10/15/11	4	1	51923	39.9	1	2014-11	11/2/10	1	MC	1	322846.93	3	3.06	3	11	3	7.38	2	0.16	2
Petaluma	CA	10/29/11	5	3	57941	74	1	2014-11	11/2/10	1	CM	2	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
Philadelphia	PA	10/6/11	207	95	1526006	83	5	2011-11	11/6/07	2	MC	1	336837.28	3	4.34	3	9	2	11.97	3	0.45	3
Phoenix	AZ	10/15/11	41	15	1445632	44	4	2011-11	8/30/11	3	CM	3	300670.23	2	2.34	2	9	2	5.6	2	0.18	2
Pittsburgh	PA	10/15/11	69	22	305704	57	2	2013-11	11/3/09	1	MC	1	218085.2	2	3	3	6	2	8.1	3	0.2	2
Pocatello	ID	10/15/11	1		54255	42	1	2013-11	11/3/09	1	MC	1	181075.86	1	1.66	1	8	2	2.43	1	0.04	1
Portland	ME	9/30/11	19	16	66194	64	1	2011-11	12/6/10	3	CM	2	188838.87	1	2.42	2	8	2	2.87	1	0.03	1
Portland	OR	10/6/11	126	111	583776	77	4	2012-11	5/20/08	1	CO	3	246669.96	2	1.73	1	11	3	5.2	2	0.11	1
Poughkeepsie	NY	10/15/11	1		32736	53.2	1	2011-11	11/6/07	2	MC	1	336021.51	3	3.27	3	0	1	9.96	3	0.06	1
Providence	RI	10/16/11	21	4	178042	83	1	2014-11	11/2/10	1	MC	2	246342.94	2	2.77	3	4	1	6.36	2	0.2	2
Raleigh	NC	10/15/11	68	7	403892	57	2	2013-10	10/11/11	2	CM	3	202590.14	1	1.76	1	9	2	4.27	1	0.19	2

Richmond	VA	10/15/11	26	55	204214	79.2	2	2012-11	11/4/08	1	CM	3	361066.38	3	3.69	3	0	1	6.99	2	0.33	3
Riverside	CA	10/15/11	16	23	303871	50	1	2012-06	11/3/09	2	CM	2	275774.92	2	1.35	1	12	3	4.31	1	0.27	2
Rochester	NY	10/1/11	14	9	210565	57	1	2014-03	3/29/11	2	MC	1	312638.38	3	3.45	3	4	1	9.64	3	0.25	2
Sacramento	CA	10/6/11	29	31	466488	59	2	2012-06	11/4/08	1	CM	3	267959.73	2	1.72	1	11	3	7.19	2	0.24	2
Salem	OR	10/10/11	7	5	154637	48	1	2012-11	11/2/10	2	CM	3	198277.06	1	1.27	1	5	2	3.36	1	0.08	1
San Antonio	TX	10/6/11	16	24	1327407	52	4	2013-06	6/13/09	1	CM	2	207993.19	1	1.52	1	6	2	5.3	2	0.49	3
San Diego	CA	10/7/11	155	243	1307402	54	4	2012-06	6/3/08	1	MC	1	285850.54	2	1.49	1	7	2	3.9	1	0.34	3
San Francisco	CA	9/17/11	195	177	805235	84	4	2011-11	1/4/11	3	MC	1	505406.38	3	2.41	2	12	3	6.67	2	0.55	3
San Jose	CA	10/2/11	35	32	945942	70	4	2014-01	6/8/10	2	CM	2	273049.29	2	1.47	1	11	3	3.39	1	0.4	3
San Leandro	CA	10/14/11	1		84950	79.7	1	2014-11	11/2/10	1	CM	2	299954.35	2	1.13	1	0	1	4.32	1	NA	NA
San Ramon	CA	10/11/11	3		72148	67.7	1	2011-11	11/3/09	2	CM	2	141376.06	1	0.8	1	0	1	0.37	1	NA	NA
Santa Ana	CA	10/22/11	26	8	324528	48	1	2012-11	11/2/10	2	CM	2	352453.02	3	1.24	1	9	2	4.05	1	0.49	3
Santa Cruz	CA	10/8/11	55	42	59946	77.8	1	2012-11	11/2/10	2	CM	2	362116.32	3	1.62	1	3	1	8.01	3	0.11	1
Santa Fe	NM	10/8/11	12	1	67947	76.8	1	2014-03	3/2/10	1	MC	2	302633.82	2	2.28	2	0	1	4.49	1	0.57	3
Santa Rosa	CA	10/15/11	29	17	167815	74	1	2012-11	11/2/10	2	CM	2	288316.29	2	1.07	1	2	1	4.06	1	0.15	2
Scranton	PA	10/16/11	4	10	76089	62.6	1	2013-11	11/3/09	1	CM	2	170852.55	1	2.02	2	0	1	2.97	1	0.01	1
Seattle	WA	10/1/11	217	97	608660	70	3	2013-11	11/3/09	1	MC	1	341734.3	3	2.12	2	10	3	6.02	2	0.24	2
Sebastopol	CA	11/5/11	11	4	7379	74	1	2012-11	11/2/10	2	CM	2	352351.27	3	2.03	2	3	1	1.08	1	0.07	1
Sonoma	CA	10/14/11	2		483878	74	3	2012-11	11/2/10	2	CM	2	269783.65	2	0.54	1	10	3	0.07	1	0.1	1
South Bend	IN	10/8/11	22	3	101168	58	1	2011-11	11/6/07	2	MC	1	280753.12	2	2.58	3	9	2	7.35	2	0.16	2
St. Louis	MO	10/1/11			319294	60	NA	2013-04	4/7/09	NA	MC	NA	429027.79	3	4.38	3	8	2	18.63	3	0.35	3

Stockton	CA	10/12/11	1		291707	55	2	2012-11	11/4/08	1	CM	2	362430.45	3	1.44	1	12	3	14.24	3	0.3	3
Syracuse	NY	10/2/11	30	1	145170	57	1	2013-11	11/3/09	1	MC	1	271496.81	2	3.43	3	9	2	8.97	3	0.08	1
Tacoma	WA	10/15/11	10		198397	55	1	2013-11	11/3/09	1	CM	2	327625.92	3	1.97	2	8	2	7.6	2	0.15	2
Tampa	FL	10/9/11	62	57	399457	53	2	2015-03	3/22/11	1	MC	1	306335.46	2	2.52	2	9	2	5.58	2	0.3	3
Toledo	OH	10/10/11	2	2	287208	65	2	2013-11	11/3/09	1	CO	3	279754.38	2	2.38	2	0	1	9.99	3	0.25	2
Trenton	NJ	10/13/11	1	4	84913	67.1	1	2014-06	6/15/10	2	MC	2	438095.46	3	4.25	3	6	2	14.23	3	0.41	3
Tucson	AZ	10/15/11	43	103	520116	52	2	2011-11	11/6/07	2	CM	2	328131.9	3	2.14	2	11	3	6.61	2	0.31	3
Tulsa	OK	10/28/11	35	97	391906	38	1	2013-11	11/10/09	1	MC	1	197549.92	1	2.12	2	8	2	10.1	3	0.22	2
Utica	NY	10/13/11	8		62235	48.6	1	2011-11	11/6/07	2	MC	1	201172.97	1	2.76	3	0	1	6.14	2	0.08	1
Virginia Beach	VA	10/15/11	2		437994	49	2	2012-11	11/4/08	1	CM	2	180559.96	1	1.87	2	12	3	1.77	1	0.16	2
Walnut Creek	CA	10/12/11	9		64173	67.7	1	2012-11	11/2/10	2	CO	3	313215.84	3	1.25	1	7	2	1.08	1	0.09	1
Washington	DC	10/1/11	95	25	601723	93	4	2014-11	11/2/10	1	MC	3	791181.39	3	6.22	3	11	3	11.61	3	0.74	3
West Palm Beach	FL	10/8/11	7	4	99919	61	1	2015-03	3/8/11	1	CO	3	430348.58	3	3.1	3	3	1	7.7	2	0.26	2
Wichita	KS	10/8/11	12	1	382368	43	2	2015-04	4/5/11	1	CM	2	170496.17	1	1.69	1	9	2	7.72	2	0.09	1
Wilmington	DE	10/15/11	3	1	70851	69.7	1	2012-11	11/4/08	1	CM	2	616123	3	4.8	3	12	3	15.68	3	0.25	2
Wilmington	NC	11/12/11	10	4	106476	49	1	2013-11	11/3/09	1	CM	2	213155.4	1	2.43	2	0	1	6.22	2	0.16	2
Worcester	MA	10/16/11	38	14	181045	55.8	1	2011-11	11/3/09	2	CM	2	199581.78	1	2.66	3	9	2	9.94	3	0.14	1
Youngstown	OH	10/15/11	8	17	66982	62	1	2013-11	11/3/09	1	MC	1	287915.96	2	2.99	3	4	1	9.24	3	0.29	3

Alphabetical Contentious (and Mixed) Performances Modeled from Protester-Initiated Gatherings

Arrests

Highest Probability: arrested, arrest, police, protesters, -year-old, _arrest_protesters_, two, one, tell, trespassing

FREX: _arrest_protesters_, pregnant, arrested, -year-old, arrest, custody, comcast, linked, sit, trespassing

Lift: _arrest_anyone, _arrest_both_, _arrest_cityname, _arrest_eight_, _arrest_eleven, _arrest_five_, _arrest_four_, _arrest_members, _arrest_nine, _arrest_p

Bank Targeting

Highest Probability: xbank, bank, protesters, branch, indep, banks, financial, california, close, district

FREX: pnc, institutions, coral, macdonald, accounts_, branch, bank, xbank, banks, financial

Lift: _arrest_bank, _arrest_nobody_, _arrest_sarah, _arrest_they_, _ask_protesters_wear, _bail_xbank, _chant_protesters_, _charge_six, _close_cityname, _close_his

City Hall Targeting

Highest Probability: city, hall, front, nov., city_tell_, lawn, xweekday, support, event, old

FREX: bagby, tech, hall, city, cityname-style, jazz, roommate, alive, duffie, jacket

Lift: alliances, angelenos, eaton, jeers, natalie, _allow_their, _ask_a, _ask_city_, _assign_the, _block_those

Curfew Disputes

Highest Probability: police, leave, p.m., park, arrested, protesters, arrest, refused, grant, tell

FREX: protesters_block_leave, grant, desks, leave, protesters_leave_, closing, refused, judge, congress, ordinance

Lift: balaklava, cta, innocent, _agree_the, _allow_dozens_continue, _allow_not/rb_the, _arrest_goodner, _arrest_hours, _arrest_its, _arrest_martinez_trying

Demonstrations

Highest Probability: demonstration, many, show, showed, man, one, indep, smaller, handful, support

FREX: demonstration, showed, demonstration_, squid, latest, goldman, smaller, show, sachs, protesters_show_

Lift: _make_at, -take, alberta, authorization, bellagio, belmont, breitbart, canada, channelside, clipboard

Encampment Activities

Highest Probability: plaza, camp, tents, set, encampment, two, center, weeks, camping, civic

FREX: protesters_camp_, _camp_protesters_, protesters_begin_camping, kitchen, tents, tents_, camp, weeks, camped, encamped

Lift: _camp_, _camp_occupy_, _camp_protesters_venting, _pitch_a, booths, city_spring_, cityname_camp_, communal, crunchy, dewitt

Labor Alliances

Highest Probability: union, groups, members, jobs, protesters, indep, employees, labor, human, international

FREX: seiu, lori, kirby, dad, jobs, union, international, human, organizations, coalition

Lift: _arrest_jenn, _arrest_n't/rb_apd_, _ask_some_, _bind_n't/rb_the, _charge_jenn, _cite_several_, _claim_a, _confuse_protesters_, _dismiss_one, _drive_many_join

Rallies

Highest Probability: occupy, rally, movement, members, home, l.., local, rallied, number, national

FREX: occupy, l.., movement, rallied, home, foreclosed, protesters_rally_, rally, homes, okc

Lift: assn., bobby, bullock, carrefour, elk, hard-hit, hull, jan., legba, lifes

Sidewalk Contestation

Highest Probability: stay, sidewalk, come, protesters_stay_, protesters_come_, eight, stood, remain, margaret, dozen

FREX: schucker, walden, dealings, speculative, wrought, margaret, protesters_stay_, attributed, stay, sidewalk

Lift: _allow_p.m., _allow_police_stay, _allow_some_do, _appear_allowing, _arrest_shucker_, _be_talk, _build_hours_, _damage_hours_, _decide_not/rb_happen_, _do_construction_

Standoffs with Riot Gear

Highest Probability: protesters, police, riot, gear, protester, move, one, deadline, cops, order

FREX: gear, rainey, deadline, riot, deputies, perimeter, standoff, tension, midnight, motorcycle

Lift: blaring, drigger, maalox, prez, uphold, _arrest_just, _arrest_protesters_blocking, _arrest_update_, _catch_protesters_, _charge_three

Traffic Battles

Highest Probability: police, protesters, tell, polices, police_tell_, said, traffic, block, pepper, spray

FREX: pepper, spray, police_tell_, blocked, polices, police, spray_, bicycles, suspects, police_be_

Lift: _arrest_suspects_, _arrest_that, _close_broad, _dismiss_that, _dismiss_video_, _free_one, _injure_several_, _join_others_, _move_suspects_, _redirect_traffic_causing

Weekday Marching

Highest Probability: cityname, xweekday, downtown, march, protested, afternoon, protesters_march_, protesting, around, evening

FREX: downtown, march, cityname, protested, xweekday, protesters_march_, afternoon, evening, protesting, sunttrust

Lift: _cite_protesters_disperse, _join_cityname, _march_the, _move_show, _show_solidarity_, 's_march_bring, afternoon_form_, ame, anti-drilling, attendees_hold_protesters_

Weekend Gatherings

Highest Probability: protesters, xweekendday, park, square, gathered, protesters_be_, gather, xpark, second, take

FREX: xpark, xweekendday, park, gather, protesters_be_, square, protesters_gather_, gathered, protesters, second

Lift: _settle_, _tell_protesters_leave, angelia, argument_flare_, camping_march_, christians, city_occupy_protesters_, city-sanctioned, cityname_have_a, comfort

Alphabetical Police Control Performances Modeled from Police-Initiated Events

Arresting Groups

Highest Probability: people, police, arrested, protesters, xpark, arrest, people_, xweekday, leave, morning, six, two, xweekendday, structure, arrests

FREX: lake, hixon, merritt, shed, people_, six, people, cuesta, structure, d.c., curtis, gentile, mcpherson, main, opposing

Lift: _charge_three, -foot-high, aboard, akard, band, boy, break-, cannon, carrefour, disruption, elderly, exposure, fitzgerald, freeways, fuel

Arresting Resisting Individuals

Highest Probability: police, one, tents, man, protesters, street, tent, take, arrested, arrest, polices, around, market, person, front

FREX: market, reserve, jessica, kneeling, laser, agitators, third, woman, man, person, chain, basillas, child, adam, man_

Lift: _dismiss_at, _take_he, _use_pepper, advisements, aluminum-frame, avenue_, cannon-like, configuration, description, devin, dirt, disassembling, ellen, fifty-five, full-time

Deadline Enforcing

Highest Probability: xpark, p.m., protesters, police, city, time, warning, arrest, deadline, xweekday, eviction, give, night, leave, anyone

FREX: grant, cesar, closure, perry, deadline, notices, closing, notices_, johnathan, p.m., anyone, permit, warning, permits, cease

Lift: _form_protesters_leave, belongings_leave_, bits, bowen_, chilling, clear-, consequences, deadline_come_, deportation, edward, frantic, iconic, immigrant, knx, mountain

Dismantling Camps

Highest Probability: protesters, police, xpark, tents, xweekday, leave, take, downtown, move, stay, remove, members, arrest, belongings, removed

FREX: downtown, tents_, villa, tents, police_tell_protesters_, members, police_stay_, protesters_leave_, remain, remove, protesters_stay_, space, pack, belongings, warned

Lift: _allow_not/rb_property_, _allow_protesters_remain, _allow_protesters_retrieve, _allow_protesters_stay, _arrest_melissa, _block_xpark_, _bring_city_clean, _bring_the, _bulldoze_an, _cite_anyone, _cite_two_, _close_downtown, _disposition_protesters_, _empty_pancho, _fine_anyone

Ordinance Enforcing

Highest Probability: city, camping, protesters, police, county, arrested, arrest, others, courthouse, xweekday, ordinance, property, enforce, .m., three

FREX: humboldt, king, subjects, -camping, prohibiting, distribution, ban, county, camping, courthouse, sanchez, ordinances, codes, fps, enforce

Lift: aresheh, carlos, chunk, conceal, craven, kris, now-banned, palmer, police_resist_city_, prohibiting, resisting/obstructing, rivera, rorey, shay, sign-bearing

Telling

Highest Probability: police, tell, said, police_tell_, xweekday, told, sgt., says, say, news, chief, lt., presence, statement, department

FREX: andy, charlie, briefing, cityname-mecklenburg, police_tell_, norwood, solano, vallejo, said, tell, capt., presence, rico, conference, police_tell_police_

Lift: armstrong, bassett, briefing, cityname-mecklenburg, fernandez, neiman, norwood, shirts, _allow_not/rb_gatherings_, _arrest_eight_, _do_monitoring_, _drive_police_, _focus_police_, _form_protesters_, _get_no

Violently Raiding

Highest Probability: police, protesters, city, polices, gas, riot_gear, unlawful, one, tear, assembly, use, cityname, hall, camp, xpark

FREX: dodger, suits, fired, beanbag, clergy, bullets, gas, affiliate, tear, projectiles, tasha, personnel, stadium, assembly, trap

Lift: _assign_, _pelt_police_, adkison, allusion, antlers, assembly_resist_police_, belts, blame, bull, cain, choya, cops_have_, countless, de-escalate, demobilize

Full 40-Topic Model for Protester-Initiated Gatherings

Topic 1 Top Words: ***Demonstrations***

Highest Probability: demonstration, many, show, showed, man, one, indep, smaller, handful, support

FREX: demonstration, showed, demonstration_, squid, latest, goldman, smaller, show, sachs, protesters_show_

Lift: _make_at, -take, alberta, authorization, bellagio, belmont, breitbart, canada, channelside, clipboard

Topic 2 Top Words:

Highest Probability: people, rally, speeches, group, conference, members, station, talk, council, alameda

FREX: clergy, blogcon, speeches, tabitha, attending, calf, protesters_talk_, foods, alameda, attended

Lift: _affirm_the, _appear_a, _arrest_julio, _block_not/rb_train, _carry_wall, _control_protesters_, _damage_indep, _damage_municipal, _damage_whole, _do_graffiti

Topic 3 Top Words:

Highest Probability: occupation, college, community, former, one, walmart, protesters, local, minnesota, student

FREX: bachmann, michele, neon, spoelstra, alan, indie, uss, burlesque, nevcal, reno

Lift: anti-mass, aresheh, bachmann, barnes, beasley, bellecourt, blended, burlesque, canyon, cello

Topic 4 Top Words:

Highest Probability: stay, sidewalk, come, protesters_stay_, protesters_come_, eight, stood, remain, margaret, dozen

FREX: schucker, walden, dealings, speculative, wrought, margaret, protesters_stay_, attributed, stay, sidewalk

Lift: _allow_p.m., _allow_police_stay, _allow_some_do, _appear_allowing, _arrest_shucker_, _be_talk, _build_hours_, _damage_hours_, _decide_not/rb_happen_, _do_construction_

Topic 5 Top Words: ***Weekday Marching***

Highest Probability: cityname, xweekday, downtown, march, protested, afternoon, protesters_march_, protesting, around, evening

FREX: downtown, march, cityname, protested, xweekday, protesters_march_, afternoon, evening, protesting, sunttrust

Lift: _cite_protesters_disperse, _join_cityname, _march_the, _move_show, _show_solidarity_, 's_march_bring,

afternoon_form_, ame, anti-drilling, attendees_hold_protesters_

Topic 6 Top Words:

Highest Probability: group, october, security, early, guards, tell, members, one, homeless, hunger

FREX: hunger, group_march_, voting, chancellor, dennis, mears, october, group, ashley, guards

Lift: -_stay_a, --book, -abandoned, adapt, advertised, alwazir, amend, batons--hand, cap, chancellor

Topic 7 Top Words: ***Arrests***

Highest Probability: arrested, arrest, police, protesters, -year-old, _arrest_protesters_, two, one, tell, trespassing

FREX: _arrest_protesters_, pregnant, arrested, -year-old, arrest, custody, comcast, linked, sit, trespassing

Lift: _arrest_anyone, _arrest_both_, _arrest_cityname, _arrest_eight_, _arrest_eleven, _arrest_five_, _arrest_four_, _arrest_members, _arrest_nine, _arrest_p

Topic 8 Top Words:

Highest Probability: street, wall, protesters, federal, market, across, anti-wall, reserve, building, street_

FREX: federal, reserve, wall, street, anti-wall, market, disjointed, street_, across, vinegar

Lift: malodorous, missives, modeled, protesters_block_market, prudential, _allow_only, _arrest_peaceful_, _back_traffic, _block_not/rb_operations, _close_avenue_

Topic 9 Top Words:

Highest Probability: protesters, police, gas, tear, cityname, frank, ogawa, olsen, xweekday, night

FREX: bean, frank, ogawa, tear, olsen, calif., fracture, gas, jamie, rounds

Lift: anda, barton, chu/staff, doernberg, dottie, fracture_, futile, gernades, grenade, krystof

Topic 10 Top Words:

Highest Probability: port, protesters, cityname, workers, terminal, p.m., tell, said, operations, trucks

FREX: port, ports, truckers, shipping, adeline, ssa, cargo, apl, hanjin, deandre

Lift: -week, arbuckle, blue-collar, cargo, cityname_close_, deandre, deprive, endanger, exports, harbors

Topic 11 Top Words:

Highest Probability: oct., since, first, common, campbell, occupying, statehouse, speech, waterfront, free

FREX: campbell, statehouse, identifies, anarchist, unitarian, universalist, since, oct., first, #occupycityname

Lift: absolute, anthems, arizonan, campbell, cass, colony, cracks, curtiss, designee, disorder

Topic 12 Top Words: ***Rallies***

Highest Probability: occupy, rally, movement, members, home, l.., local, rallied, number, national

FREX: occupy, l.., movement, rallied, home, foreclosed, protesters_rally_, rally, homes, okc

Lift: assn., bobby, bullock, carrefour, elk, hard-hit, hull, jan., legba, lifes

Topic 13 Top Words:

Highest Probability: outside, building, protesters, headquarters, office, main, children, morning, st., front

FREX: stumpf, headquarters, outside, office, banker, max, parents, children, rubright, sits

Lift: _charge__, _charge_not/rb_protesters_, _close_banks, _evict_school, _leave_police_, _speak_protesters_, _think_the, -percenters, -percenters_march_pacing, ''_come_indep_

Topic 14 Top Words:

Highest Probability: streets, hundreds, crowd, estimated, intersection, thousands, manhattan, xplaza, morning, xbridge

FREX: anniversary, streets, estimated, lower, hundreds, manhattan, irwin, two-month, thousands, streets_

Lift: dubinsky, kickoff, ninty-nine, party-like, _assign_police_lead, _block_bullhorns, _claim_police_act, _close_third, _estimate_protesters_hour, _force_xbank_close

Topic 15 Top Words:

Highest Probability: protesters, driver, convention, police, tell, car, said, center, one, three

FREX: pearce, coal, hartwell, export, driver, convention, heidi, steve, walter, suntrust

Lift: _allow_cityname, _allow_her, _allow_protesters_continue, _allow_protesters_leave, _allow_workers_, _arrest_steve, _arrest_three_obey, _assess_no, _assess_the, _assign_protesters_

Topic 16 Top Words:

Highest Probability: protest, begin, day, began, corporate, last, greed, started, month, weekend

FREX: protest_begin_, protest, greed, protesters_begin_, corporate, began, begin, month, upwards, day

Lift: case--case, edt, group_march_protest, group_protest_brutality_, hartzell, lummas, mourn, occupants, protest_go_, protesters_begin_their

Topic 17 Top Words:

Highest Probability: protesters, small, school, teachers, political, rallies, westlake, district, members, economic

FREX: institute, laney, brittlebank, halsey, merry, telegram, westlake, rallies, christmas, holiday

Lift: archways, barbara, blvd, boy, brokers, bushwick, cardenas, cerritos, charter, cross-representation

Topic 18 Top Words:

Highest Probability: way, protests, church, protesters, way_, sixth, mellon, made, make, cityname

FREX: cynthia, construction, ingram, buyers, cousin, church, gardens, mellon, sixth, bny

Lift: bny-mellon, catholic, cooking, ingram, olive, protests_begin_, quite, _arrest_her, _ask_participants_snap,

_claim_the

Topic 19 Top Words: ***Encampment Activities***

Highest Probability: plaza, camp, tents, set, encampment, two, center, weeks, camping, civic

FREX: protesters_camp_, _camp_protesters_, protesters_begin_camping, kitchen, tents, tents_, camp, weeks, camped, encamped

Lift: _camp_, _camp_occupy_, _camp_protesters_venting, _pitch_a, booths, city_spring_, cityname_camp_, communal, crunchy, dewitt

Topic 20 Top Words:

Highest Probability: state, capitol, courthouse, grounds, building, protesters, steps, front, others, capital

FREX: troopers, tennessee, capitol, grounds_, j.j., grounds, state, demonstrate, courthouse, hopkins

Lift: alexandra, alora, apologized, auctioned, bayless, bitten, chairs_, cook, elaine, fans

Topic 21 Top Words:

Highest Probability: arrests, avenue, four, make, made, mall, reported, arrests_, michigan, p.m.

FREX: southwest, crabtree, arrests_, arrests, o'neill, portion, king, michigan, reported, valley

Lift: anti-camping, brisk, buren, burson, crabtree, csu, dontae, early-evening, ehrlich, emily

Topic 22 Top Words: ***Traffic Battles***

Highest Probability: police, protesters, tell, polices, police_tell_, said, traffic, block, pepper, spray

FREX: pepper, spray, police_tell_, blocked, polices, police, spray_, bicycles, suspects, police_be_

Lift: _arrest_suspects_, _arrest_that, _close_broad, _dismiss_that, _dismiss_video_, _free_one, _injure_several_, _join_others_, _move_suspects_, _redirect_traffic_causing

Topic 23 Top Words:

Highest Probability: protesters, another, police, one, said, protester, woman, photographer, back, civil

FREX: embarcadero, attackers, another, sharp, slashed, freelance, photographer, brandon, ones, disobedience

Lift: abdomen, eve, graber, pen-like, protesters_try_take, retrieve, accelerating, amash, annoyed, cagle

Topic 24 Top Words: ***Bank Targeting***

Highest Probability: xbank, bank, protesters, branch, indep, banks, financial, california, close, district

FREX: pnc, institutions, coral, macdonald, accounts_, branch, bank, xbank, banks, financial

Lift: _arrest_bank, _arrest_nobody_, _arrest_sarah, _arrest_they_, _ask_protesters_wear, _bail_xbank, _chant_protesters_, _charge_six, _close_cityname, _close_his

Topic 25 Top Words:

Highest Probability: night, police, protesters, josh, park, three, tell, hours, time, n't

FREX: chavez, cesar, harkinson, robertson, rollins, fellows, josh, night, buncombe, vance

Lift: chavez, rollins, _admit_trucks, _allow_n't/rb_city_, _allow_protesters_protest, _anger_last, _answer_city_, _carry_bill, _close_city_, _close_police_

Topic 26 Top Words:

Highest Probability: part, government, pindep, plaza, hotel, singer, party, obama, rain, county

FREX: singer, part-time, blues, mcdaniel, songwriter, ellas, government, hennepin, downpour, fundraising

Lift: _allow_n't/rb_a, _arrest_ellas, _arrest_joseph, _arrest_october, _arrest_protesters_prove, _arrest_these, _block_three, _cite_these, _crowd_bags, _disband_barricades

Topic 27 Top Words: ***City Hall Targeting***

Highest Probability: city, hall, front, nov., city_tell_, lawn, xweekday, support, event, old

FREX: bagby, tech, hall, city, cityname-style, jazz, roommate, alive, duffie, jacket

Lift: alliances, angelenos, eaton, jeers, natalie, _allow_their, _ask_a, _ask_city_, _assign_the, _block_those

Topic 28 Top Words:

Highest Probability: move, week, moved, participants, protesters_move_, last, meet, met, event, roughly

FREX: fleming-salopek, furloughed, jill, publicly-owned, week, protesters_move_, participants, moved, effigy, recently

Lift: accion, bemused, books, buffet, busloads, centered, clerk-recorder, colgate, communities, condi

Topic 29 Top Words: ***Curfew Disputes***

Highest Probability: police, leave, p.m., park, arrested, protesters, arrest, refused, grant, tell

FREX: protesters_block_leave, grant, desks, leave, protesters_leave_, closing, refused, judge, congress, ordinance

Lift: balaklava, cta, innocent, _agree_the, _allow_dozens_continue, _allow_not/rb_the, _arrest_goodner, _arrest_hours, _arrest_its, _arrest_martinez_trying

Topic 30 Top Words:

Highest Probability: people, several, hundred, people_, dozens, least, side, one, crowd, including

FREX: sabeghi, daniele, erville, choi, goodstal, people, salmun, hundred, several, people_

Lift: _anger_protesters_, _arrest_fifteen, _arrest_indep, _arrest_then_, _assign_not/rb_both, _bail_n't/rb_h, _bring_police_control, _carry_at, _charge_city_, _charge_n't/rb_both

Topic 31 Top Words:

Highest Probability: protesters, lot, fence, two, legislative, hearing, property, mike, vacant, colorado

FREX: reapportionment, o'kelly, korzen, fence, dirt, lot, hearing, chain-link, colorado, mike

Lift: korzen, _allow_not/rb_your, _arrest_anna, _arrest_bradley, _arrest_not/rb_a, _attack_some, _block_no, _bring_dinner_, _climb_city_, _continue_march

Topic 32 Top Words:

Highest Probability: police, stephen, benavides, protesters, video, tell, arrest, said, one, arrested

FREX: stephen, benavides, planter, benavides_, hollis, ixchel, aguilar, cuesta, video, servant

Lift: benavides_stay_, body-slammed, dedrick, improper, limb, padierna, penny, planter, sexual, syrian

Topic 33 Top Words:

Highest Probability: bridge, university, protesters, traffic, river, one, campus, boulevard, bus, peaceful

FREX: bridge_, bridge, river, university, bus, lasalle, boulevard, amitai, heller, magick

Lift: _tell_each, -pack, alton, amitai, backups, bobbe, bohemia, charlestown, connecting, cornelius

Topic 34 Top Words:

Highest Probability: protesters, dozen, chanting, chant, young, two, slogans, store, protesters_chant_, peace

FREX: dack, decker, maupin, excluded, garden, mass., loudly, dozen, alfred, causa

Lift: a_yell_, aisles_, alfred, antonio, award, beebe, beebe_tell_, blazers, brutalizing, causa

Topic 35 Top Words: ***Labor Alliances***

Highest Probability: union, groups, members, jobs, protesters, indep, employees, labor, human, international

FREX: seiu, lori, kirby, dad, jobs, union, international, human, organizations, coalition

Lift: _arrest_jenn, _arrest_n't/rb_apd_, _ask_some_, _bind_n't/rb_the, _charge_jenn, _cite_several_, _claim_a, _confuse_protesters_, _dismiss_one, _drive_many_join

Topic 36 Top Words:

Highest Probability: signs, supporters, tent, indep, holding, carry, south, group, chants, carrying

FREX: ari, sterling, salina, post-standard, clean-, dissatisfaction, marino, signs, douglas, magnell

Lift: actively, hand-held, pfenning, post-standard, reilly, _move_ari, ability, akbar, ari, backpacks_be_

Topic 37 Top Words:

Highest Probability: police, protesters, building, .m., broadway, fire, vacant, general, people, strike

FREX: traveler, san, explosions, fire, launchers, travelers, aid, fires, pablo, broadway

Lift: _promise_those, antagonize, bang_be_, black-clad, blitzkrieg, bop, break-, building_take_, calm_settle_, cement

Topic 38 Top Words: ***Weekend Gatherings***

Highest Probability: protesters, xweekendday, park, square, gathered, protesters_be_, gather, xpark, second, take

FREX: xpark, xweekendday, park, gather, protesters_be_, square, protesters_gather_, gathered, protesters, second

Lift: _settle_, _tell_protesters_leave, angelia, argument_flare_, camping_march_, christians, city_occupy_protesters_, city-sanctioned, cityname_have_a, comfort

Topic 39 Top Words:

Highest Probability: new, york, meeting, exchange, council, stock, protesters, concert, public, veterans

FREX: york, jersey, hip-hop, nicholas, dorsey, hop, proper, new, exchange, concert

Lift: _accuse_others_, _allow_not/rb_protesters_conduct, _allow_not/rb_tents_, _allow_protesters_sleep, _bill_what_, _bring_the, _catch_cops_physical, _cite_, _cite_-year-old, _continue_challenge

Topic 40 Top Words: ***Standoffs with Riot Gear***

Highest Probability: protesters, police, riot, gear, protester, move, one, deadline, cops, order

FREX: gear, rainey, deadline, riot, deputies, perimeter, standoff, tension, midnight, motorcycle

Lift: blaring, drigger, maalox, prez, uphold, _arrest_just, _arrest_protesters_blocking, _arrest_update_, _catch_protesters_, _charge_three

Full 15-Topic Model Results for Police-Initiated Events

Topic 1 Top Words:

Highest Probability: police, protesters, citations, polices, arrest, cite, night, cityname, arrested, xweekendday, issued, city, cited, pepper, xweekday

FREX: garcia, steffen, rene, szayer, commerce, nicholas, relevant, scotney, outlining, rosemary, virato, citations, chamber, joseph, laws

Lift: _give_, _relate_the, -tent, account_, acoustical, alley, asheland, bleary-eyed, car--car, chat, cheering, code_arrest_, code_be_arrest, construct, conte

Topic 2 Top Words:

Highest Probability: cityname, protesters, group, arrested, arrest, trespassing, dozen, indep, xweekendday, arrests, eight, public, safety, city, protest

FREX: jonathan, meador, mary, foreclosed, glen, hacktivist, kazmi, ayesha, leroy, griffin, gov., belcik, okc, safety, marijuana

Lift: _arrest_jonathan, _charge_all_leave, alesandra, all_leave_the, alternative, amons, beasely, belcik_tell_, bellos, biohazards, bless, brutally, colleagues, conjunction, cottontown

Topic 3 Top Words:

Highest Probability: state, protesters, xpark, xstate, patrol, police, -year-old, xweekday, troopers, capitol, arrested, district, nine, arrest, oct.

FREX: capitol, ian, pearl, common, gregory, lorenzo, gunter, meighan, monroe, hunter, sur, affluent, xstate, state, de-unlawfully

Lift: _accuse_gregory, _allow_n't/rb_protesters_do, _allow_protesters_be, _arrest_faith, _arrest_gregory, _arrest_ian, _arrest_national, _arrest_nine_, _arrest_police_obey, _arrest_protesters_leave, _arrest_thirty-eight, _arrest_thomas, _arrest_twenty-five, _arrest_twenty-nine, _arrest_twenty-three

Topic 4 Top Words: ***Ordinance Enforcing***

Highest Probability: city, camping, protesters, police, county, arrested, arrest, others, courthouse, xweekday, ordinance, property, enforce, .m., three

FREX: humboldt, king, subjects, -camping, prohibiting, distribution, ban, county, camping, courthouse, sanchez, ordinances, codes, fps, enforce

Lift: aresheh, carlos, chunk, conceal, craven, kris, now-banned, palmer, police_resist_city_, prohibiting, resisting/obstructing, rivera, rorey, shay, sign-bearing

Topic 5 Top Words:

Highest Probability: police, center, protesters, civic, plaza, people, move, two, cityname, tent, camp, .m., area, one, city

FREX: barrel, --work, places, shaunn, civic, center, cartwright, cleaning, luke, protestersground, remains, walker, extinguisher, irrigation, skillets

Lift: _accuse_most_, _allow_dog, _allow_not/rb_cooking, _allow_not/rb_police_have, _allow_not/rb_those, _allow_police_, _allow_those, _arrest, _arrest_joining_, _arrest_police_disperse, _block_some, _build_some, _camp_protesters_, _carry_three_, _charge_richard

Topic 6 Top Words:

Highest Probability: protesters, police, xweekday, week, last, cityname, put, raid, former, force, arrest, xpark, marine, smoking, arrested

FREX: marine, suspect, cigarette, steuart, carotid, prysner, tall, viejo, husband--wife, pounds, spear, former, smoking, bed, spitting

Lift: cigarette, marine, _admit_protesters_, _allow_never/rb_protesters_erect, _arrest_never/rb_protesters_, _arrest_who_, _awake_eyewitness, _charge_anti-war, _charge_many, _charge_mike, _dislodge_protesters_, _dismiss_vita, _disperse, _empty_protesters, _evict_

Topic 7 Top Words: ***Dismantling Camps***

Highest Probability: protesters, police, xpark, tents, xweekday, leave, take, downtown, move, stay, remove, members, arrest, belongings, removed

FREX: downtown, tents_, villa, tents, police_tell_protesters_, members, police_stay_, protesters_leave_, remain, remove, protesters_stay_, space, pack, belongings, warned

Lift: _allow_not/rb_property_, _allow_protesters_remain, _allow_protesters_retrieve, _allow_protesters_stay, _arrest_melissa, _block_xpark_, _bring_city_clean, _bring_the, _bulldoze_an, _cite_anyone, _cite_two_, _close_downtown, _disposition_protesters_, _empty_pancho, _fine_anyone

Topic 8 Top Words: ***Violently Raiding***

Highest Probability: police, protesters, city, polices, gas, riot_gear, unlawful, one, tear, assembly, use, cityname, hall, camp, xpark

FREX: dodger, suits, fired, beanbag, clergy, bullets, gas, affiliate, tear, projectiles, tasha, personnel, stadium, assembly, trap

Lift: _assign_, _pelt_police_, adkison, allusion, antlers, assembly_resist_police_, belts, blame, bull, cain, choya, cops_have_, countless, de-escalate, demobilize

Topic 9 Top Words: ***Deadline Enforcing***

Highest Probability: xpark, p.m., protesters, police, city, time, warning, arrest, deadline, xweekday, eviction, give, night, leave, anyone

FREX: grant, cesar, closure, perry, deadline, notices, closing, notices_, johnathan, p.m., anyone, permit, warning, permits, cease

Lift: _form_protesters_leave, belongings_leave_, bits, bowen_, chilling, clear-, consequences, deadline_come_, deportation, edward, frantic, iconic, immigrant, knx, mountain

Topic 10 Top Words: ***Telling***

Highest Probability: police, tell, said, police_tell_, xweekday, told, sgt., says, say, news, chief, lt., presence, statement, department

FREX: andy, charlie, briefing, cityname-mecklenburg, police_tell_, norwood, solano, vallejo, said, tell, capt., presence, rico, conference, police_tell_police_

Lift: armstrong, bassett, briefing, cityname-mecklenburg, fernandez, neiman, norwood, shirts, _allow_not/rb_gatherings_, _arrest_eight_, _do_monitoring_, _drive_police_, _focus_police_, _form_protesters_, _get_no

Topic 11 Top Words:

Highest Probability: city, police, encampment, occupy, camp, hall, xweekday, raid, early, polices, empty, plaza, area, morning, cleared

FREX: calvin, nola, milam, spring, _evict_protesters_, encampment, encampment_, l.., occupy, empty, nypd, hall, raided, raids, cleared

Lift: _add_, _affirm_that_, _arrest_almost, _arrest_approximately, _arrest_around, _arrest_arthur, _arrest_calvin, _arrest_hundreds_, _arrest_journalists_, _arrest_lawn, _arrest_your_, _ask_city_clear,

_be_six, _bound_the, _cancel_preparation

Topic 12 Top Words: ***Arresting Resisting Individuals***

Highest Probability: police, one, tents, man, protesters, street, tent, take, arrested, arrest, polices, around, market, person, front

FREX: market, reserve, jessica, kneeling, laser, agitators, third, woman, man, person, chain, basillas, child, adam, man_

Lift: _dismiss_at, _take_he, _use_pepper, advisements, aluminum-frame, avenue_, cannon-like, configuration, description, devin, dirt, disassembling, ellen, fifty-five, full-time

Topic 13 Top Words:

Highest Probability: police, arrests, street, make, xweekendday, xpark, protesters, cityname, structures, arrests_, bradley, watch, made, passageway, russell

FREX: passageway, russell, mat, winter, sink, weeks-old, southwest, bradley, arrests_, structures, resisted_, fourteen, sycamore, weather, watch

Lift: _arrest_bradley, _disband_, anti-structure, arrestees_confront_potential, arrestees_evict_, artwork, balanced, city_set_a, cloud, college-aged, constitution, dana, degree, dorothy, emaining

Topic 14 Top Words:

Highest Probability: protesters, police, protester, one, arrest, justin, tell, move, rodriguez, arrested, five, bridges, upper, represents, bob

FREX: justin, rodriguez, bridges, represents, bob, o'grady, chair, momentoff, shelby, george, maria, uriah, curb, vanessa, bridges_

Lift: _accuse_five, _allow_not/rb_city_see, _allow_protesters_use, _arraign_the, _arrest_bob, _arrest_not/rb_joe, _arrest_o'grady_, _arrest_wbz-tv, _ask_nobody_leave, _assign_anything_, _attack_protester, _blind_eli, _block_protesters_, _block_traffic_, _carry_joe

Topic 15 Top Words: ***Arresting Groups***

Highest Probability: people, police, arrested, protesters, xpark, arrest, people_, xweekday, leave, morning, six, two, xweekendday, structure, arrests

FREX: lake, hixon, merritt, shed, people_, six, people, cuesta, structure, d.c., curtis, gentile, mcpherson, main, opposing

Lift: _charge_three, -foot-high, aboard, akard, band, boy, break-, cannon, carrefour, disruption, elderly, exposure, fitzgerald, freeways, fuel

Equations for Structural Topic Model Estimates and Figures

The intuition behind structural topic modeling is fully described in Chapter 3. Structural topic modeling uses the same Latent Dirichlet Allocation algorithm as used by better-known "topic modeling." (Readers interested in more detail about the Latent Dirichlet Allocation algorithm used in Structural Topic Modeling should see Roberts, Stewart, and Tingley 2014.) Once topics are discovered by that algorithm, just as in traditional LDA topic modeling, output also includes estimates of the distribution of topics across documents and the aggregate corpus. *Structural* topic models allow researchers to *also* use "meta-data" variables describing documents as independent variables predicting or explaining the prevalence of topics across the corpus. The intuition: variables that describe the documents (for examples, where they are from or when they were written) can help explain why a topic was more or less prevalent in a document or a class of documents. Therefore, hypotheses predicting topic prevalence (a continuous variable) as a function of independent variables describing documents, may be tested using general linear regression models.

Here, I specify the models used to estimate and then plot almost all of the Figures of this dissertation. Each relies on one of the two following equations (A or B, below) where $\gamma =$ the prevalence of a topic in the corpus; $\tau =$ the number of days between an event described by the document (text unit) and the establishment of the local Occupy camp; β_1, $\beta_2, \beta_3, \ldots \beta_n$ represent coefficients for variables describing the cities that documents (text units) are about (i.e., number of political liberals in a city, the number of power centers in its political structure, the instability of its political system, and the capacity and culture of its police force); and ε is an error term.

(A) $$\gamma = \tau(x) + \beta_1(x) + \beta_2(x) + \beta_3(x) + \cdots \beta_n(x) + \varepsilon$$

(B) $$\gamma = \tau(x) + \beta_1 \times \tau(x) + \beta_2(x) + \beta_3(x) + \cdots \beta_n(x) + \varepsilon$$

The simplest initial models of Chapter 4 estimated, predicted, and displayed topic prevalence for the corpus of text units describing protester-initiated contentious gatherings. These models estimated $\beta_1, \beta_2, \beta_3, \ldots \beta_n$ using equation A. Figures 4.2 – 4.7 then predict and display γ across (the duration of Occupy campaings) τ while holding $\beta_1, \beta_2, \beta_3$ (describing the number of Obama supporters, number of power centers, and stability of political alliances) at their means.

Figures 4.8 – 5.8 display the predicted effects of $\beta_1, \beta_2, \beta_3, \ldots \beta_n$ on γ over τ. These figures also begin with the coefficient estimates of eqution A. However, each Figure displays a predicted γ across three different values of one β (e.g. β_1) for τ while holding the other β coefficients (e.g. β_2, β_3) at their means.

Chapter 5 argues that this approach to estimation and prediction is inadequate because it does not allow researchers to observe how different values of β effect γ differently through time, τ. Therefore, the remaining figures of the dissertation use equation B to interact β with τ, while holding other β coefficients at their means.

Figures 5.9 – 6.14 all display predicted values of γ across three different values of one β for τ while holding the other β coefficients (e.g. β_2, β_3) at their means. And since these models include the interaction term $\beta_1 \times \tau$, predictions of γ are not simply raised or lowered vertically as in Figures 4.8 – 5.8. The shapes of the γ prediction lines for each value of β across τ move with the data.

The only thing that differs across Figures 5.9 – 6.14 are the specifications of the main β of interest, and the control variables $\beta_n, \beta_m, \ldots \beta_o$. Each figure's legend clearly labels the

values of the β of interest. And the Figure's notes indicate which predicted γ (topic/performance prevalence) is being displayed in each Panel of the Figure. For all contentious performance figures (Figures 5.9 – Figures 5.20), the same political opportunity structure variables are used: Number of Obama Supporters, Number of Power Centers, and Political Instability Score. These are each continuous variables rendered ordinal for estimation and visualization purposes. Their operationalizations are fully described in Chapter 5. All of these POS variables are used in each model, but only one is interacted with τ in each model – the β of interest.

Figures 6.1-6.14 use a larger array of control variables. Since both city-level POS variables *and* variables describing police departments are likely to affect police-initiated control performances, all these variables are used in each model. Again, as in Figures 4.8 – 5.20, all but one of these variables are held at their means when predicting γ for display in figures. And as in Figures 5.9 – 5.20, one β of interest is interacted with τ as described in Equation B, then predictions of γ are displayed for each value of the β of interest across τ.

Supplementary Figures for Topic Modeling of Control Performances

Figure 1, below, shows the proportion of each control performance in the corpus of text units describing police-initiated events. Figure 2 shows topic correlations for police control performances. One can see that, especialy when compared to protesters' contentious performances, police control performances are often decided independently of one another. They are not closely linked.

Figure A.1 – Topic Proportions for all Control Performances

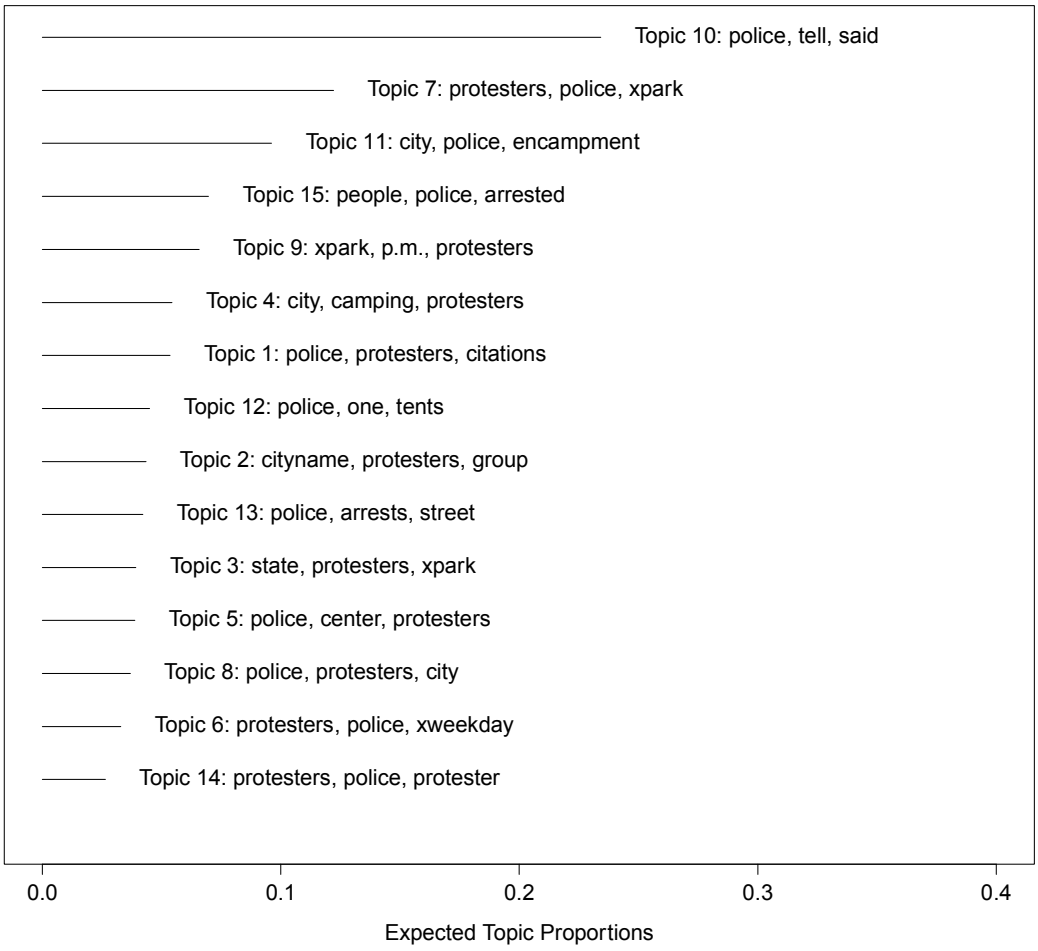
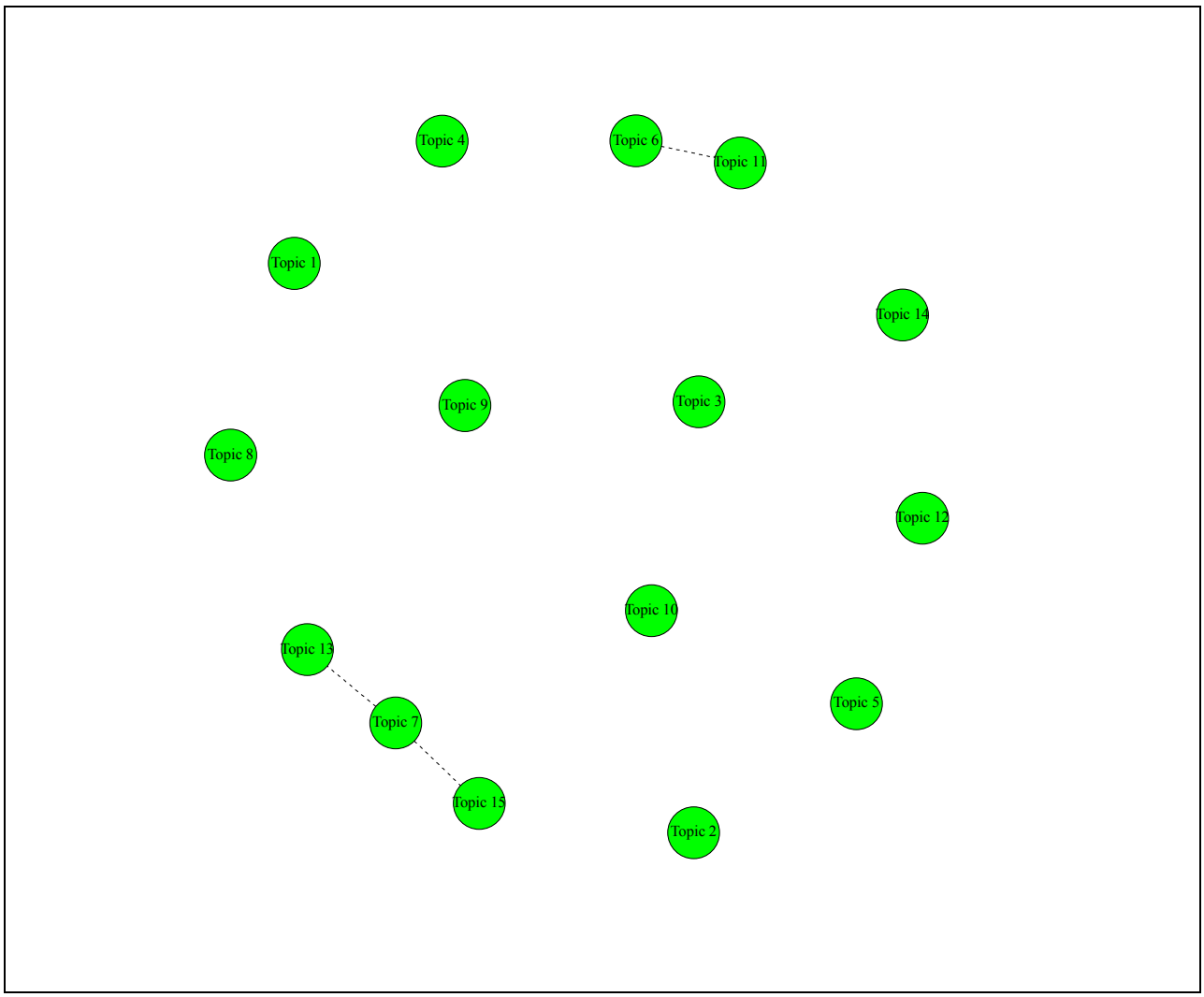


Figure A.2 – Topic Corrleations for Control Performances



Description and protocol of City/Encampment identification procedure

Our team identified the universe of all U.S. Occupy encampments that met the following criteria:

- Had 2 or more tents for 2 or more nights
- Were NOT a college campus encampment
- Were NOT equal to or smaller than (secondary to) a college encampment in the same city

We did this by applying the following search protocol to two different lists of cities – one from the U.S. Census, and a second from the "Occupy Directory" http://directory.occupy.net/, which sought to document all Occupy encampments around the world.

1. Google search occupy [city name]

2. If occupy in a city has a different name than the city name (e.g. occupy Marin instead of occupy Healdsburg), make sure google searches in steps 3 -- 5 use the non-standard name (e.g. occupy Marin)

3. Google search "[ocupy name] encampment'" using either city name or alternate occupy name

4. Google search "[occupy name] camp" using either city name or alternate occupy name

5. Google search "[occupy name] tent" using either city name or alternate occupy name

6. Determine whether the city fits our criteria of selection by scanning the articles resulting from searches in steps 3 --5. If none of the searches provide solid evidence that the city had a camp with 2 or more tents for 2 or more nights, that city will not be in our sample. If the city had a camp, but it was on a college campus, it will NOT be in our sample. If the city had a camp that was the same size as, or smaller than, a camp on a college campus located in the same city, it will NOT be in our sample.

We used two different lists to ascertain which camps fit our criteria because it was impractical to check every city in the Census list of U.S. cities. While most large cities had an encampment, many smaller cities and towns did not. Sampling from the list was not appropriate because we have sought to analyze the full universe of Occupy Encampments and government/police responses. The Occupy Directory proved to be a great resource. Its team was especially permissive in its criteria of

inclusion. Many 'encampments' in the Occupy Directory were little more than facebook pages that organized occasional actions. Therefore, while it was necessary to remove most cities in their list from our dataset, their permissiveness gave us confidence that we, indeed, identified every encampment in the U.S.

Protocol for article gathering

This research PROTOCOL is designed to systematize our article collection and saving process to ensure that we are all putting in uniform and maximally efficient effort as we hunt down articles to include in our dataset for further coding.

GENERAL NOTES:
Only research for one city at a time. (Complete the article/blog search process for that city before moving on to perform the process for any other city. It might seem like it is easier to search for two cities at once in some cases. But it actually is not a good practice because the only way to ensure that you've thoroughly and adequately gathered articles for a city or town is to go through this whole protocol from start to finish anyway. Do not worry about sending articles to your peers. It is more efficient to let others find documents for their respective cities.
We are searching for and saving articles that have information on events (not mere opinion about the movement or information about potential upcoming events) – events that took place between September 1, 2011 and December 31, 2011. (This is an intentional narrowing of the scope of the project.) Some more recent articles, even those written as late as August 2012 ((when we stopped the practice of collecting more contemporary articles)), may be useful and you may include them. But if the article refers only to events occurring after 2011, we are not interested. Do not save it. If it is someone's opinion about the movement without referencing any events, don't save it.
DETAILED STEPS:
1. Find out the name of the encampment you are looking for. Usually it will be 'Occupy [city name],' but sometimes it will be different. Do a Google Search to find out what it is called. The name will be important for your keyword searches.

2. Go to Lexis/Nexis. Get there by going to the Berkeley library site,'s 'Electronic Resources' page http://www.lib.berkeley.edu/find/types/electronic_resources.html and typing 'Lexis' into the search bar. Click on 'LexisNexis Academic':

a. Click on the 'News' button in the left column
b.Click on 'All News'

c. Search For: "[name of encampment (from Step 1 e.g. "occupy Houston"]"; Specify Date: "Date is between… 9/01/2011 and 12/31/2011"; Select Source: By Type: "US Newspapers & Wires"

d. Click red 'Search' button

e. When the results come up, choose Sort: oldest to newest

f. Skim through the articles or just their headlines enough so that you know if they are important – i.e., if they have any factual information about Occupy events and camps (including op-eds sometimes). If they are important, click the box to the left of the article title (which will tag the article). As you scan through the entire search results (which may be several pages long), these boxes will remain checked.

g.Once you have scanned the entire list and checked the boxes for all relevant articles, click the blue "View tagged articles" button

h.Find the 'save' icon in the upper right. It looks like an old computer disk (the kind that no one uses anymore.) Click on that and follow the instructions to save all articles in the search into a single Word document on your computer. Uncheck the boxes for 'Cover Page,' 'Each Document on New Page,' and 'search terms in bold.' (There should be no boxes checked.) Click 'Download.'

i. Copy and paste the contents of this Word document into GoogleDrive/EVENTS/[Cityname] as a Google doc. Give it the name 'Lexis_[OccupyName]' Note: If it is a big file (with more than 50 or so articles), you may have to copy and paste 50 articles or so at a time from the word Doc into the Google Doc. (Google Drive will crash if, for instance, you try to copy and paste the whole 396-article Lexis dump from Occupy Oakland.).

3. Google Search to find out the names of local newspapers and TV and radio stations with websites.

a. Go to Google and type '[city name, state] local news'

b. Notice the 'Search Tools' button near the middle of the screen, second line, to the right of the 'More' button. Click on it. It will show you that you are searching from a particular city. (Google weights its searches based on where you sit in geographic space.) Change the locations so that Google will believe you are in the city on which you are performing your search. (This will automatically update the search results, though it is common for nothing to change on the first page of results.)

c. Note the names and web addresses of the local newspapers, radio stations, TV stations, etc. in a Google document saved in the GoogleDrive/EVENTS/[Cityname]. Name that file '[cityname]_local_sources.'

4. For each local news source perform a Google Search using '[localnewspaperwebsiteaddress.com]: Occupy [cityname]' e.g. "khou.com: occupy

houston" (When you use '[website address]: ' before your search term, you are telling Google to do its thing on that site only. This is usually the best way to search a site's archives.) ALSO, be sure to limit the temporal range of your search. Click on 'show search tools,' then 'Custom Range,' then set the date range from September 1, 2011 – Dec 31,2011.

a. Review and consider saving all of the articles in the Google results until you get to three consecutive Google results pages whose links are 90% or more totally irrelevant.

b. Save all articles covering events. (Review criteria above.) Copy and paste the text of the articles into a Google doc. Usually it is best to view the article in 'Printer-Friendly' mode and then just copy and paste it into the Google doc. You can place multiple articles into this Google document – as many articles as the news source has.

c. Save the Google document in the DecidingForce/ARTICLES/[Cityname folder as '[name of news source]' for example, "Independent_News_Pensacola"

d. Repeat these steps for each local news source. This will take some time because many cities have multiple radio stations, TV news networks, major papers, and smaller papers.

Protocol for article formatting

We are preparing each article for coding. The goal is to reduce the article to basic text. These are the steps to do just that, as follows:

1. Open the article. If it is in Microsoft Word, or an html document, simply right-click the article and use the "Open With" from the drop-down menu and choose Google Docs. This will automatically convert the file to a Google document. If it is already a Google Doc, just open it as it is.

2. Hit Ctrl + A to choose all the text.

3. In your edit bar above, left-click Normal text and choose "Normal text" from the drop-down menu. Repeat this action. I don't know why we have to do it twice, but we do.

4. Keeping the whole text field chosen, next go over to the Line spacing button and click it. Choose Line spacing 1.0

5. Next, choose Arial as your font.

6. Choose 12 pt. pica, just to the right of Arial.

7. Bold the text, then unbold it.

8. Italicize the text, then unitalicize it.

9. Underline the text, then hit that button again.

10. Choose Text color and indicate black.

11. At text background color, choose “None”

Now we are ready to prepare the article for coding. Each article needs the following:

+*+*

Most recent date of article

So, if the date of the article is say, 10-17-11, but was updated on 10-18-11, the top of the article will look like this,

+*+*

10-18-11

12. Leave a space under the date of the article, and then single space all the information underneath that regarding the publication information. Just use whatever information is there.

13. Leave a space before the body of the text. Throughout the article, delete any photos that occur, but leave the photo caption if it indicates a codeable tidbit. Otherwise, you can delete the photo caption as well.

Also, if the article is using a paragraph space after every sentence, eliminate those spaces as well.

14. At the end of the article, leave a space between it and any further publication information, if there is any.

15. If there is another article in the file, leave a space and type in the +*+* with the date as indicated above.

16. If it is the last article (or only article) in that particular file, just simply close it and move on to the next.

You will come across bits and pieces in some of these articles that are superfluous to the text. If, and only if, they are completely irrelevant to the Occupy article, you may delete these. An example would be where the journal has put in links for printing, or sharing, or perhaps links to other articles.

Don't hesitate to ask questions. Look at the articles I've already done (A's and most of the B's) and see what it looks like.

You can call or text me at xxx-xxx-xxxx with any questions at all. Remember: the only stupid question is the one you didn't ask!

Description and protocol for text unit identification

ARTICLE Text Unit Identification PROTOCOL

1. Open the 'DECIDING FORCE' folder and then the 'ARTICLES' folder. You'll see that there are a lot of articles organized by City. Go to your assigned city and find any document.

2. Note whether the file has a square blue 'Google Doc' icon or a 'W' icon. If it has a square icon, open the document and proceed to step 3. If the icon is a 'W' you will need to open the document and convert it to a Google Document. To do this, click on the name of the document. Google viewer will show you what it looks like. Click on the blue 'Open' button at the bottom right of the screen. (This will open the document for limited functionality in Google.) Then, go to the 'File' menu in the upper left of your window and toggle over 'Open with...,' then click on 'Open with Google Docs.' Now, the document has been converted to a Google Doc that will allow you to edit the document, especially to color code the text.

3. Open the Color Coding Scheme document in a separate window. You'll want to open as a Word document (as opposed to a Google Doc) so you can view it in outline view. If you "show" Level2 of the outline only, you'll see all the color categories on one page. If you "show" Level 3, you'll get some more detail on the things we are looking for in each color-code category.

4. Place your initials and the coding scheme version number under the date of each article you code. You should see articles arranged like so:

+*+* ← These symbols denote a new article

11-3-11 ← On the very next line, the article date

[enter text here] ← Your initials and version # (e.g. 'NBA, v23')

ARTICLE HEADLINE

Article text about occupy events, camps, city actions, etc. Article text about occupy events, camps, city actions, etc. Article text about occupy events, camps,

city actions, etc. Article text about occupy events, camps, city actions, etc. … [End of article.]

+*+*

11-3-11 ← it should look like this once you're coding it

NBA, v23 (yes, that comma and space are important)

ARTICLE HEADLINE

Article text about occupy events, camps, city actions, etc. Article text about occupy events, camps, city actions, etc. Article text about occupy events, camps, city actions, etc. Article text... etc. … … [End of article.]

5. Color code all articles in the Google document according to your training and the coding scheme. If an article describes more than one police-, protester-, or unknown-initiated event, you need to not only color-code those events, but also change their font to italics (if the second such event), underline (if the third), strikethrough (if the fourth), etc. Same goes for multiple camps. See the bottom of the color-coding scheme for the ways that you can indicate 5, 6, or 7 events

6. Once you have color-coded all articles in the Google Doc, move on to the next Google Doc and repeat steps 1 thru 5.

Hand-coding Scheme for TUA Identification

Police-Initiated EVENT

- ****EVENT TYPE**
  - **Raid of camp**
  - **Showdown at the camp (i.e. police show up creating a tense situation that might lead to arrests)**
  - **Warnings or Threats at the camp or through the media (i.e. police inform campers about impending enforcement of curfew orders with intention of coercing a change in behavior)**
  - **Surveillance of camp**
  - **Surveillance of individual protest organizers away from the camp**
  - **Arrests of protest organizers away from any protest-initiated EVENT or CAMP**
  - **Undercover infiltration**
  - **Propaganda camping against occupy (perhaps through social media)**
  - **"Snatch 'n' grab" arrests at camp – arrest of individual people (especially key organizers, at the camp, even with a warrant)**
  - **Space-taking (i.e. establishing of perimeter barricades or fences or lines of police for containment or strategic incapacitation purposes)**
  - **Statement (to public and/or occupy) of REFUSAL to enforce an eviction order or permit expiration or some other law coming from city**
  - **New! Notable Non-event e.g. intentional non-enforcement – Everyone is expecting police to do one thing, then they don't do it**
  - **Official statement from police chief, or high-level officer or official spokesperson**
    - ***Code the act of speaking, writing an open letter, op-ed or press release in blue along with the name and/or title of the official spokesperson***
      - Code the content of the speech act according to the rest of the coding scheme

- **CROWD COMPOSITION (How many…?)**
  - **Protesters**
  - **Protesters from different cities**
    - ***From where***
    - ***How many***
  - **Counterprotesters**
  - **Anarchists**
  - **Union Members**
  - **Media**
  - **General Indicator of Diversity of…**
    - ***Ages***
    - ***Classes***

Ethinicities%
…%

Police-presence-characteristics-
Riot-gear-
Horses-
Urban-Assault-Vehicles-
Brandishing-weapons-
Skirmish-lines-

Actions-of-police-or-protesters-that-occurred-during-the-event-
Who-
Police%
Protester%
Counter5protester%
Anarchist%
How%many%
Tag%
Estimate(
What-
Punch%
Kick%
Push/shove%
To-Whom-
Police%
Protester%
Counter5protester%
Anarchist%
Media%
How%many%
Tag%
Estimate(
Time-

Sequence-order-
Date/Time-tag–
START-
END-
-

Protester'Initiated-EVENT-
Date/Time-tag–
START-
END-
-
****EVENT-TYPE-**
Establishing-a-camp-(will-often-look-like-a-rally)-

Voluntary Dissolution of a camp—not directly resulting from a police-initiated raid
Moving a camp to a new location (will often look like a march)—
NOTE: implies the dissolution of the previous camp, so only count it as a move if the other camp is dissolving in the process. If the old camp survives while some of the people move to a new location, the move to a new location should simply be coded as the establishment of a new camp
March/Parade (unless protesters do rally/demonstration-type stuff at the start point or end point, a march from A to B should just be counted as one event.)
Rally/Demonstration
NEW!! Disrupting an on-going event of the perceived 1% (e.g. conservative politician's speech at a convention/country club, city council meeting ONLY if no intention to negotiate or make a proposal, etc.)
Strike
Divestment action (e.g. moving money from banks to credit unions)
Blocking Action
Sidewalk
Street
Public transportation
Airport
Shipping port
Strategic violence
Kidnapping
Assassination
Bombing
Assault
Strategic sabotage
Pre-planned vandalism
Pre-planned arson

PERMITTED?
From when
Till when
With what conditions

CROWD-COMPOSITION (How many people…?)
Diversity of…
Ages
Classes
Ethinicities
…
Police

- Protesters-
- New!-Protesters-from-other-cities-
  - *Which%city?%*
- Counterprotesters-
- Anarchists-
- Media-
- Union-Members-
- Religious-leaders-or-communities-
- Occupy-the-Hood-Folks-
- Other-ALLIED-Groups-

Police-presence-characteristics-

- Riot-gear-
- Horses-
- Urban-Assault-Vehicles-
- Brandishing-weapons-
- Skirmish-lines-

Actions-of-police-or-protesters-that-occurred-during-the-event-

- Who-
  - *Police%*
  - *Protester%*
  - *Counter5protester%*
  - *Anarchist%*
  - *How%many%*
    - Tag%
      - *Estimate(*
- What-
  - *Punch%*
  - *Kick%*
  - *Push/shove%*
  - *Invite%arrest%*
  - *And%many%more…%*
- To-Whom-
  - *Police%*
  - *Protester%*
  - *Counter5protester%*
  - *Anarchist%*
  - *Media%*
  - *How%many%*
    - Tag%
      - *Estimate(*
- Time-
- Sequence-order-

Evidence-of-Community-Support-

High-Level Orientation Document for CoreNLP Coreference Resolution and ClausIE SVO Extraction

Getting Started with CoreNLP

Download

You can download pre-compiled versions of CoreNLP to run it without having to compile from source just so you can try it out and see how it works.

We also need to download the models using this in a pom.xml in the top level of the CoreNLP directory and run it with:

mvn

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-
v4_0_0.xsd">
 <modelVersion>4.0.0</modelVersion>
 <groupId>org.decidingforce</groupId>
 <artifactId>df-parser</artifactId>
 <packaging>jar</packaging>
 <version>1.0-SNAPSHOT</version>
 <name>df-parser</name>
 <url>http://maven.apache.org</url>
 <dependencies>
   <dependency>
     <groupId>junit</groupId>
     <artifactId>junit</artifactId>
     <version>3.8.1</version>
     <scope>test</scope>
   </dependency>
   <dependency>
     <groupId>edu.stanford.nlp</groupId>
     <artifactId>stanford-corenlp</artifactId>
     <version>3.5.0</version>
   </dependency>
   <dependency>
     <groupId>edu.stanford.nlp</groupId>
     <artifactId>stanford-corenlp</artifactId>
```

```
    <version>3.5.0</version>
    <classifier>models</classifier>
  </dependency>
 </dependencies>
</project>
```

Run

We are running it using a command line like:

```
java -Xmx2g -cp classes:models
edu.stanford.nlp.pipeline.StanfordCoreNLP -annotators
cleanxml,tokenize,ssplit,pos,lemma,ner,parse,dcoref -file
/path/to/annotatedinput.xml
```

The main Stanford documentation describes Usage of CoreNLP in more detail.

Once you become familiar with simply running it from the pre-compiled binaries then you may want to look at the source.

Source code

We are using the code from github for Stanford's CoreNLP source, so you can git clone that to get started looking at the code.

To make our modifications we have to fully recompile it and I can go over that with you on Wednesday when we get together.

Compiling from source

Latest git repo: `git@github.com:stanfordnlp/CoreNLP.git`

Clone with: `git clone --depth 1 git@github.com:stanfordnlp/CoreNLP.git`

```
cd CoreNLP ; ant compile
export CLASSPATH=/path/to/CoreNLP/classes
```

Example Deciding Force Data

Hand-annotated input

Our input data would look like the following (color-added for emphasis/clarity; text in **bold** needs coreference resolution):

<doc city="San Francisco" date="2015-03-29" articleid="3">A great university called <tua type=”A” id=”7”>Stanford is located in California. <tua type=”B” id=”3”> Nick has an old college buddy who works as a researcher at **its** medical school.</tua> </tua> California is in the USA. <tua type=”A” id=”24”> **He** wants to visit soon.</tua></doc>

Coreference resolution stage

A pass through coreference resolution might transform this input data into output like:

<doc city="San Francisco" date="2015-03-29" articleid="3">A great university called <tua type=”A” id=”7”> Stanford is located in California. <tua type=”B” id=”3”> Nick has an old college buddy who works as a researcher at **<dcoref original="its">Stanford’s</dcoref>** medical school.</tua> </tua> California is in the USA. <tua type=”A” id=”24”>**<dcoref original=”He”>Nick</dcoref>** wants to visit soon.</tua></doc>

ClausIE stage

the text below is struck out for the moment because the output format is subject to change, potentially significantly, so writing a preprocessor for the next step right now might be subject to quite a bit of changes.

~~Then ClausIE will be applied to the coreference transformed text to generate:~~

~~<doc city="San Francisco" date="2015-03-29" articleid="3">A great university called <tua type=”A” id=”7”> **<clauses>**Stanford **<svo S="Stanford" V=”located” O=”California”>**is located**</svo>** in California.**</clauses>** <tua type=”B” id=”3”>**<clauses>**Nick **<svo S=”Nick” V=”has” O=”old college buddy”>**has**</svo>** an old college buddy who **<svo S=”old college buddy” V=”works” O=”researcher”>**works**</svo>** as a researcher at <dcoref original="its">Stanford’s</dcoref> medical school.**</clauses>**</tua> </tua> California is in the USA. <tua type=”A” id=”24”>**<clauses>**<dcoref original=”He”>Nick</dcoref> **<svo S=”Nick” V=”visit” O=””>**wants to visit**</svo>** soon.**</clauses>**</tua></doc>~~

Preprocessing XML in Python for Actor Dictionary

Given the output from the ClausIE stage above, then using Python lxml must be first scanned for TUAs, then the following would be extracted and output in a format suitable for input to the topic modeling (probably some kind of flat, CSV format easily read into an R dataframe) , which might look something like the following for an article that contains three distinct TUAs; 2 type “A” TUAs and 1 type “B” TUA. Here is what a transformation from XML to a linear format would like BEFORE applying the Actor Dictionary:

city	date	article id	tua type	tua id	s	v	o
San Francisco	2015-03-29	3	A	7	Stanford	located	California
San Francisco	2015-03-29	3	A	7	Nick	has	old college buddy
San Francisco	2015-03-29	3	A	7	old college buddy	works	researcher
San Francisco	2015-03-29	3	B	3	Nick	has	old college buddy

San Francisco	2015-03-29	3	B	3	old college buddy	works	researcher
San Francisco	2015-03-29	3	A	24	Nick	visit	

IMPORTANT NOTE: If you review the XML output from ClausIE, you'll notice that tua B,3 is contained entirely within tua A,7, so the output above shows duplicate SVOs for because the SVOs in B,3 also apply to A,7 because it is contained inside it.

Applying Actor Dictionary

Then it should be simple to apply the actor dictionary such as this:

Nick = Protester
Stanford = City
researcher = University Employee

to generate output like so:

city	date	article id	tua type	tua id	s	v	o
San Francisco	2015-03-29	3	A	7	City	located	California
San Francisco	2015-03-29	3	A	7	Protester	has	old college buddy
San Francisco	2015-03-29	3	A	7	Employee	works	University
San Francisco	2015-03-29	3	B	3	Protester	has	University Employee
San Francisco	2015-03-29	3	B	3	old college buddy	works	University
San Francisco	2015-03-29	3	A	24	Protester	visit	

Customizing CoreNLP for Deciding Force

Write an xml module (modules are called "annotators" in CoreNLP)

The first thing that needs to be replaced/modified in the source is the cleanxml annotator which simply removes xml tokens from the document, but we want to grab the annotations on each span and add it to the annotator pipeline.

Borrowing from GATE

It may be possible to re-use ideas or code from GATE (or examples), starting with their XmlDocumentHandler.java.

One of the GATE authors says:

> Keeping Your Annotations: It looks as if your documents are essentially XML. This is good as GATE will happily load XML documents. The text of the document is used as the document content while your annotations will end up in the "Original markups" annotation set. Any annotations (regardless of which set they are in can be kept and exported again later).
>
> (Coreference Resolution: You'll find there is a coreference PR as part of the base ANNIE application. It doesn't alter the document content though. What you'll end up with is annotations that reference each other to encode the coreference chain.)
>
> Exporting Data: Given that you want to actually edit the document content you'll probably need to do some work to produce documents in the form you want. Fortunately with the current SVN version this is easier than it used to be. What you'll need to do is produce an instance of the gate.DocumentExporter class. There are two exporters in the Linguistic_Simplifier plugin which do something similar (they remove/replace text during export) and would probably be easy to adapt to your specific use case.

What to do after collecting xml attributes from hand-coded annotations

There is more that needs to be done after reading the xml and preserving annotations:

Our text data pipeline requires that we carry human-generated, *xml-based* annotations of words (examples in color below) through a text pipeline that includes coreference resolution *and* SVO extraction via ClausIE (an annotator/module of CoreNLP).

A summary of the areas of CoreNLP code and the approach is suggested by some folks at Stanford on the mailing list.

> **GOAL:**
> **Carry human annotations of words through the Stanford CoreNLP.**

APPROACH:
Create two classes that implement the abstract class edu.stanford.nlp.pipeline.Annotator. (for example, see CleanXmlAnnotator, EntityMentionsAnnotator or POSTaggerAnnotator)

[POS == Parts of Speech]

Add the annotators to the available pools of annotators. (See StanfordCoreNLP class (lines like "pool.register(STANFORD_POS, AnnotatorFactories.posTag(properties, annotatorImplementation));").) You will need to change AnnotatorImplementations and AnnotatorFactories classes. (Note: There's also a way to set the annotators set dynamically using the properties file.) (Also note: when you do read_id , you can set a class for that label with the value of the id.

Add a class to CoreAnnotations and then
for each token (of type CoreLabel)
Do something like:
l.set(CoreAnnotations.CustomID.class, tokenid).

Print token id using l.get(CoreAnnotations.#CustomID.class)

AMPLIFYING NOTES AND DISCUSSION
-Each token is an instance of CoreLabel, which is a Map from (Class) keys to some information you provide. You can stick any extra attributes into that map you want, and they will be carried forward through the pipeline. (They won't print by default in our output formats, but you can access them and print them.)
-A key has to be a class that implements CoreAnnotation<V>, where V is the type of the value. It doesn't have to be defined in CoreAnnotations. You can define the class anywhere, but you can look at CoreAnnotations to see what they should look like.

ClausIE: SVO extractor

Open Source license and permissions to use unpublished code

---------- Forwarded message ----------
From: **Luciano Del Corro** <corrogg@mpi-inf.mpg.de>
Date: Tue, Apr 7, 2015 at 5:07 AM
Subject: Re: ?Using ClausIE in the open?
To: Nicholas Adams <nickbadams@berkeley.edu>

Cc: Rainer Gemulla <rgemulla@mpi-inf.mpg.de>

Hi Nicholas

1) You can put the code there, no problem. Just put a link in the README file to our webpage.

2) I think so, our license is http://creativecommons.org/licenses/by-sa/3.0/

cheers
Luciano

Code Enacting CoreNLP Coreference Resolution and ClausIE SVO Extraction

```
diff --git a/_COPYRIGHTS_ b/_COPYRIGHTS_
new file mode 100644
index 0000000..7fb9b6f
--- /dev/null
+++ b/_COPYRIGHTS_
@@ -0,0 +1,8 @@
+Portions of the code reproduced below include original code
+written for the Deciding Force project and is interspersed with
+code verbatim, dervied, or modified based on the Stanford CoreNLP
+codebase which is goverened by the GNU GPL v2 license
+<http://www.gnu.org/licenses/old-licenses/gpl-2.0.en.html>, as
+well as the ClausIE code adapted from Luciano Del Corro
+<corrogg@mpi-inf.mpg.de> with permission to use under the
+CC-BY-SA 3.0 license <http://creativecommons.org/licenses/by-sa/3.0/>
diff --git a/BUILD.md b/BUILD.md
new file mode 100644
index 0000000..789a57f
--- /dev/null
+++ b/BUILD.md
@@ -0,0 +1,24 @@
+Some useful commands to keep track of used in the build process:
+
+
+```
+sudo gpg --keyserver keyserver.ubuntu.com --recv-keys C2518248EEA14886
+sudo gpg --export --armor C2518248EEA14886 | sudo apt-key add -
+sudo add-apt-repository -y ppa:webupd8team/java
+sudo add-apt-repository -y ppa:openjdk-r/ppa
+sudo apt-get update
+sudo apt-get install -y openjdk-8-jdk openjdk-8-jre openjdk-8-jre-headless oracle-java8-installer oracle-java8-set-default ant make
maven2 libmaven-ant-tasks-java screen byobu tmux
```

```
+sudo update-alternatives --config java
+sudo update-alternatives --config javac
+sudo apt-get autoremove --purge -y
+```
+
+Old direct-maven installation is no longer necessary since we're using libmaven-ant-tasks-java
+
+```
+mvn archetype:generate -DgroupId=org.decidingforce -DartifactId=df-parser -DarchetypeArtifactId=maven-archetype-quickstart -
DinteractiveMode=false
+mvn -DskipTests package
+mvn -DskipTests -Xlint:unchecked package
+mvn clean install -U
+mvn dependency:copy-dependencies
+```
diff --git a/build.xml b/build.xml
index b8fc811..435d2ac 100644
--- a/build.xml
+++ b/build.xml
@@ -1,7 +1,13 @@
 <!-- JavaNLP core build file -->

-<project name="core" default="compile" basedir=".">
-
+<project name="core" default="compile" basedir="." xmlns:artifact="antlib:org.apache.maven.artifact.ant">
+  <record name="build.log" loglevel="verbose" action="start" />
+  <path id="maven-ant-tasks.classpath" path="lib/maven-ant-tasks.jar" />
+  <typedef resource="org/apache/maven/artifact/ant/antlib.xml"
+           uri="antlib:org.apache.maven.artifact.ant"
+           classpathref="maven-ant-tasks.classpath" />
+
+  <property name="build.compiler.emacs" value="on"/>
   <property name="build.path"     value="${basedir}/classes" />
   <property name="source.path"    value="${basedir}/src" />
   <property name="doc.path"       value="${basedir}/doc" />
@@ -82,6 +88,15 @@
     </for>
   </target>

+  <artifact:dependencies pathId="dependency.classpath">
+    <dependency groupId="junit" artifactId="junit" version="3.8.2" scope="test"/>
+    <dependency groupId="org.codehaus.jackson" artifactId="jackson-mapper-asl" version="1.9.13" scope="runtime"/>
+    <dependency groupId="net.sf.jopt-simple" artifactId="jopt-simple" version="4.4" scope="runtime"/>
+    <dependency groupId="edu.stanford.nlp" artifactId="stanford-corenlp" version="3.5.0" classifier="models" scope="runtime"/>
+    <dependency groupId="xom" artifactId="xom" version="1.2.5" scope="runtime"/>
+    <dependency groupId="xalan" artifactId="xalan" version="2.7.0" scope="runtime"/>
+  </artifact:dependencies>
+
   <target name="compile" depends="classpath"
         description="Compile core sources">
     <echo message="${ant.project.name}" />
@@ -97,6 +112,7 @@
           fork="true"
           memorymaximumsize="2g"
           includeantruntime="false">
+      <classpath refid="dependency.classpath" />
       <classpath refid="classpath" />
     <!-- <compilerarg value="-Xmaxerrs"/>
          <compilerarg value="20"/> -->
@@ -107,6 +123,7 @@
       <compilerarg value="-Xlint:finally"/>
       <compilerarg value="-Xlint:path"/>
       <compilerarg value="-Xlint:try"/>
+      <!-- <compilerarg value="-Xlint:unchecked"/> -->
 <!--
       <compilerarg value="-Xlint:deprecation"/>
       <compilerarg value="-Xlint:dep-ann"/>
@@ -143,7 +160,7 @@
     </junit>
```

```
  </target>

-  <target name="itest" depends="classpath,compile"
+    <target name="itest" depends="classpath,compile"
         description="Run core integration tests">
    <echo message="${ant.project.name}" />
    <junit fork="yes" maxmemory="8g" printsummary="off" outputtoformatters="false" forkmode="perTest" haltonfailure="true">
@@ -202,6 +219,66 @@
    </java>
  </target>

+  <target name="run-corpus-whole" depends="compile,classpath">
+    <property name="data.workset" value="${data.path}/workset" />
+    <property name="data.outpath" value="${data.path}/svoset" />
+    <mkdir dir="${data.outpath}" />
+    <exec executable="data/regenerate-listings.sh"/>
+    <java classname="edu.stanford.nlp.pipeline.StanfordCoreNLP" fork="true" jvm="java">
+      <jvmarg value="-Xmx4g"/>
+      <jvmarg value="-Xdebug"/>
+      <arg value="-props"/>
+      <arg value="df.properties"/>
+      <arg value="-filelist"/>
+      <arg value="data/corpus-whole.list"/>
+      <arg value="-outputDirectory"/>
+      <arg value="${data.outpath}"/>
+      <classpath>
+        <path refid="dependency.classpath" />
+              <path refid="classpath"/>
+              <pathelement path="${build.path}"/>
+      </classpath>
+    </java>
+  </target>
+
+  <target name="run-clausie" depends="compile,classpath">
+    <!-- <java classname="${run.class}" fork="true" jvm="java"> -->
+    <java classname="clausie.ClausIE" fork="true" jvm="java">
+      <jvmarg value="-Xmx2g"/>
+      <jvmarg value="-Xdebug"/>
+      <!-- <arg value="-v"/> -->
+      <arg value="-c"/>
+      <arg value="resources/clausie.conf"/>
+      <arg value="-f"/>
+      <!-- <arg value="data/df-subset/clausie-test.txt"/> -->
+      <arg value="data/df-subset/simple.txt"/>
+      <classpath>
+        <path refid="dependency.classpath" />
+              <path refid="classpath"/>
+              <pathelement path="${build.path}"/>
+      </classpath>
+    </java>
+  </target>
+
+  <target name="run-df" depends="compile,classpath">
+    <!-- <java classname="${run.class}" fork="true" jvm="java"> -->
+    <java classname="org.decidingforce.DecidingForce" fork="true" jvm="java">
+      <jvmarg value="-Xmx2g"/>
+      <jvmarg value="-Xdebug"/>
+      <!-- <arg value="-v"/> -->
+      <arg value="-c"/>
+      <arg value="resources/clausie.conf"/>
+      <arg value="-f"/>
+      <!-- <arg value="data/df-subset/clausie-test.txt"/> -->
+      <arg value="data/df-subset/simple.txt"/>
+      <classpath>
+        <path refid="dependency.classpath" />
+              <path refid="classpath"/>
+              <pathelement path="${build.path}"/>
+      </classpath>
+    </java>
```

```
+  </target>
+
   <!-- Same as "run," except causes the VM to wait until debugger is attached -->
   <!-- See http://nlp.stanford.edu/javanlp/did_you_know/eclipse_debug.html for example -->
   <target name="run-debug" depends="classpath">
diff --git a/df.properties b/df.properties
new file mode 100644
index 0000000..103501f
--- /dev/null
+++ b/df.properties
@@ -0,0 +1,86 @@
+annotators = tokenize, cleanxml, ssplit, pos, lemma, ner, parse, dcoref
+
+# A true-casing annotator is also available (see below)
+#annotators = tokenize, ssplit, pos, lemma, truecase
+
+# A simple regex NER annotator is also available
+# annotators = tokenize, ssplit, regexner
+
+#Use these as EOS punctuation and discard them from the actual sentence content
+#These are HTML tags that get expanded internally to correct syntax, e.g., from "p" to "<p>", "</p>" etc.
+#Will have no effect if the "cleanxml" annotator is used
+#ssplit.htmlBoundariesToDiscard = p,text
+
+#
+# None of these paths are necessary anymore: we load all models from the JAR file
+#
+
+#pos.model = /u/nlp/data/pos-tagger/wsj3t0-18-left3words/left3words-distsim-wsj-0-18.tagger
+## slightly better model but much slower:
+##pos.model = /u/nlp/data/pos-tagger/wsj3t0-18-bidirectional/bidirectional-distsim-wsj-0-18.tagger
+
+# If you set ner.model, you can name any arbitrary model you want.
+# The models named by ner.model.3class, ner.model.7class, and
+# ner.model.MISCclass are also added in the order named.
+# Any of the ner.model properties can be a comma separated list of names,
+# in which case each of the models in the comma separated list is added.
+#ner.model = ...
+#ner.model.3class = /u/nlp/data/ner/goodClassifiers/all.3class.distsim.crf.ser.gz
+#ner.model.7class = /u/nlp/data/ner/goodClassifiers/muc.distsim.crf.ser.gz
+#ner.model.MISCclass = /u/nlp/data/ner/goodClassifiers/conll.distsim.crf.ser.gz
+
+#regexner.mapping = /u/nlp/data/TAC-KBP2010/sentence_extraction/type_map_clean
+#regexner.ignorecase = false
+
+#nfl.gazetteer = /scr/nlp/data/machine-
reading/Machine_Reading_P1_Reading_Task_V2.0/data/SportsDomain/NFLScoring_UseCase/NFLgazetteer.txt
+#nfl.relation.model =
/scr/nlp/data/ldc/LDC2009E112/Machine_Reading_P1_NFL_Scoring_Training_Data_V1.2/models/nfl_relation_model.ser
+#nfl.entity.model =
/scr/nlp/data/ldc/LDC2009E112/Machine_Reading_P1_NFL_Scoring_Training_Data_V1.2/models/nfl_entity_model.ser
+#printable.relation.beam = 20
+
+#parser.model = /u/nlp/data/lexparser/englishPCFG.ser.gz
+#parser.flags = -retainTmpSubcategories
+
+#srl.verb.args=/u/kristina/srl/verbs.core_args
+#srl.model.cls=/u/nlp/data/srl/trainedModels/englishPCFG/cls/train.ann
+#srl.model.id=/u/nlp/data/srl/trainedModels/englishPCFG/id/train.ann
+
+#coref.model=/u/nlp/rte/resources/anno/coref/corefClassifierAll.March2009.ser.gz
+#coref.name.dir=/u/nlp/data/coref/
+#wordnet.dir=/u/nlp/data/wordnet/wordnet-3.0-prolog
+
+#dcoref.demonym = /scr/heeyoung/demonyms.txt
+#dcoref.animate = /scr/nlp/data/DekangLin-Animacy-Gender/Animacy/animate.unigrams.txt
+#dcoref.inanimate = /scr/nlp/data/DekangLin-Animacy-Gender/Animacy/inanimate.unigrams.txt
+#dcoref.male = /scr/nlp/data/Bergsma-Gender/male.unigrams.txt
+#dcoref.neutral = /scr/nlp/data/Bergsma-Gender/neutral.unigrams.txt
```

```
+#dcoref.female = /scr/nlp/data/Bergsma-Gender/female.unigrams.txt
+#dcoref.plural = /scr/nlp/data/Bergsma-Gender/plural.unigrams.txt
+#dcoref.singular = /scr/nlp/data/Bergsma-Gender/singular.unigrams.txt
+
+#whether or not to print singleton entities
+#output.printSingletonEntities = false
+output.printSingletonEntities = false
+outputExtension = .jsonl
+replaceExtension = true
+
+# This is the regular expression that describes which xml tags to keep
+# the text from.  In order to on off the xml removal, add cleanxml
+# to the list of annotators above after "tokenize".
+#clean.xmltags = .*
+# A set of tags which will force the end of a sentence.  HTML example:
+# you would not want to end on <i>, but you would want to end on <p>.
+# Once again, a regular expression.
+# (Blank means there are no sentence enders.)
+#clean.sentenceendingtags =
+# Whether or not to allow malformed xml
+#clean.allowflawedxml
+# clean.sectiontags = "p"
+# clean.sectionAnnotations = normalized=span[type-id]
+clean.sentenceendingtags = p
+clean.docAnnotations = docID=article[metadata]
+# clean.sectionAnnotations = docID=article[city], dfID=article[srcpath], doctype=article[city], docdate=article[date],
docsourcetype=article[annotators]
+# clean.sectionAnnotations = df-type-id=span[type-id], df-type=span[type]
+clean.tokenAnnotations = df-type-id=span[type-id-color]
+
+continueOnAnnotateError = true
+tparse.maxlen = 70
diff --git a/pom.xml b/pom.xml
new file mode 100644
index 0000000..c5a67be
--- /dev/null
+++ b/pom.xml
@@ -0,0 +1,51 @@
+<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
+  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
+  <modelVersion>4.0.0</modelVersion>
+  <groupId>org.decidingforce</groupId>
+  <artifactId>df-parser</artifactId>
+  <packaging>jar</packaging>
+  <version>1.0-SNAPSHOT</version>
+  <name>df-parser</name>
+  <url>http://maven.apache.org</url>
+  <build>
+    <resources>
+      <resource>
+        <directory>src/</directory>
+      </resource>
+    </resources>
+  </build>
+  <dependencies>
+    <dependency>
+      <groupId>org.codehaus.jackson</groupId>
+      <artifactId>jackson-mapper-asl</artifactId>
+      <version>1.9.13</version>
+    </dependency>
+    <dependency>
+      <groupId>junit</groupId>
+      <artifactId>junit</artifactId>
+      <version>3.8.1</version>
+      <scope>test</scope>
+    </dependency>
+    <dependency>
+      <groupId>net.sf.jopt-simple</groupId>
+      <artifactId>jopt-simple</artifactId>
```

```
+    <version>4.4</version>
+   </dependency>
+   <dependency>
+            <groupId>xom</groupId>
+            <artifactId>xom</artifactId>
+            <version>1.2.5</version>
+   </dependency>
+   <dependency>
+    <groupId>edu.stanford.nlp</groupId>
+    <artifactId>stanford-corenlp</artifactId>
+    <version>3.5.0</version>
+   </dependency>
+   <dependency>
+    <groupId>edu.stanford.nlp</groupId>
+    <artifactId>stanford-corenlp</artifactId>
+    <version>3.5.0</version>
+    <classifier>models</classifier>
+   </dependency>
+  </dependencies>
+</project>
diff --git a/src/clausie/AdvclIndexedConstituent.java b/src/clausie/AdvclIndexedConstituent.java
new file mode 100644
index 0000000..e5db38f
--- /dev/null
+++ b/src/clausie/AdvclIndexedConstituent.java
@@ -0,0 +1,43 @@
+package clausie;
+
+import java.util.List;
+import java.util.Set;
+
+import edu.stanford.nlp.ling.IndexedWord;
+import edu.stanford.nlp.semgraph.SemanticGraph;
+
+public class AdvclIndexedConstituent extends IndexedConstituent{
+
+            private IndexedWord mark;
+            private List<Clause> clauses;
+
+
+     private AdvclIndexedConstituent() {
+
+     }
+
+
+     public AdvclIndexedConstituent(SemanticGraph semanticGraph, IndexedWord root,
+                         Type type, List<Clause> clauses) {
+          super(semanticGraph, root, type);
+          this.setClauses(clauses);
+     }
+
+
+            public AdvclIndexedConstituent(SemanticGraph semanticGraph, IndexedWord root,
+                                    Set<IndexedWord> additionalVertexes,
+                                    Set<IndexedWord> excludedVertexes, Type type, List<Clause> clauses) {
+                         super(semanticGraph, root, additionalVertexes, excludedVertexes, type);
+                         this.setClauses(clauses);
+            }
+
+            /** Returns the clauses derived from the constituent. */
+            public List<Clause> getClauses() {
+                         return clauses;
+            }
+
+            /** Sets the clauses derived from the constituent. */
+            public void setClauses(List<Clause> clauses) {
+                         this.clauses = clauses;
+            }
+}
```

```
diff --git a/src/clausie/CCompIndexedword.java b/src/clausie/CCompIndexedword.java
new file mode 100644
index 0000000..06a528e
--- /dev/null
+++ b/src/clausie/CCompIndexedword.java
@@ -0,0 +1,48 @@
+package clausie;
+
+import java.util.List;
+import java.util.Set;
+
+import edu.stanford.nlp.ling.IndexedWord;
+import edu.stanford.nlp.semgraph.SemanticGraph;
+
+public class CCompIndexedword extends StructuredConstituent {
+
+
+	private List<Clause> clauses;
+
+
+	public CCompIndexedword(SemanticGraph semanticGraph, IndexedWord root,
++ 			Type type, List<Clause> clauses) {
+		super(semanticGraph, root, type);
+		this.setClauses(clauses);
+	}
+
+
+	public CCompIndexedword(SemanticGraph semanticGraph, IndexedWord root,
++ 			Set<IndexedWord> additionalVertexes,
+ 			Set<IndexedWord> excludedVertexes, Type type, List<Clause> clauses) {
+		super(semanticGraph, root, additionalVertexes, excludedVertexes, type);
+		this.setClauses(clauses);
+	}
+
+	/** Returns the clauses derived from the constituent. */
+	public List<Clause> getClauses() {
+		return clauses;
+	}
+
+	/** Sets the clauses derived from the constituent. */
+	public void setClauses(List<Clause> clauses) {
+		this.clauses = clauses;
+	}
+
+
+ 	/**
+ 	 * @param args
+ 	 */
+ 	public static void main(String[] args) {
+ 			// TODO Auto-generated method stub
+
+ 	}
+
+}
diff --git a/src/clausie/ClausIE.java b/src/clausie/ClausIE.java
new file mode 100644
index 0000000..a36edff
--- /dev/null
+++ b/src/clausie/ClausIE.java
@@ -0,0 +1,529 @@
+package clausie;
+
+import java.io.DataInput;
+import java.io.DataInputStream;
+import java.io.FileInputStream;
+import java.io.FileOutputStream;
+import java.io.IOException;
+import java.io.InputStream;
+import java.io.OutputStream;
```

```
+import java.io.PrintStream;
+import java.io.StringReader;
+import java.util.ArrayList;
+import java.util.Collection;
+import java.util.HashMap;
+import java.util.List;
+import java.util.Map;
+
+import clausie.Constituent.Flag;
+import clausie.JavaUtils.MapUtil;
+import joptsimple.OptionException;
+import joptsimple.OptionParser;
+import joptsimple.OptionSet;
+import edu.stanford.nlp.io.EncodingPrintWriter.out;
+import edu.stanford.nlp.ling.CoreLabel;
+import edu.stanford.nlp.ling.IndexedWord;
+import edu.stanford.nlp.parser.lexparser.LexicalizedParser;
+import edu.stanford.nlp.parser.lexparser.LexicalizedParserQuery;
+import edu.stanford.nlp.pipeline.ParserAnnotatorUtils;
+import edu.stanford.nlp.process.CoreLabelTokenFactory;
+import edu.stanford.nlp.process.Morphology;
+import edu.stanford.nlp.process.PTBTokenizer;
+import edu.stanford.nlp.process.TokenizerFactory;
+import edu.stanford.nlp.semgraph.SemanticGraph;
+import edu.stanford.nlp.semgraph.SemanticGraphFactory;
+import edu.stanford.nlp.trees.Tree;
+import edu.stanford.nlp.util.ScoredObject;
+
+public class ClausIE {
+		Tree depTree;
+		SemanticGraph semanticGraph;
+		List<ScoredObject<Tree>> trees;
+
+	List<Clause> clauses = new ArrayList<Clause>();
+
+	boolean kparse = false;
+	int k = 10;
+
+	double bestScore;
+	Tree bestDT;
+	SemanticGraph bestSemanticGraph;
+	List<Clause> bestClauses;
+
+	List<Proposition> propositions = new ArrayList<Proposition>();
+	Map<Proposition, Double> scoredPropositions = new HashMap<Proposition, Double>();
+
+	PropositionGenerator propositionGenerator;
+
+	Options options;
+
+	private LexicalizedParser lp;
+	private TokenizerFactory<CoreLabel> tokenizerFactory;
+	private LexicalizedParserQuery lpq;
+
+	// Indicates if the clause processed comes from an xcomp constituent of the
+	// original sentence
+	boolean xcomp = false;
+	private Morphology morphology;
+
+	// -- construction
+	// ------------------------------------------------------------------------
+
+	public ClausIE(Options options) {
+		this.options = options;
+		this.propositionGenerator = new DefaultPropositionGenerator(this.options);
+	}
+
+	public ClausIE() {
+		this(new Options());
```

```
+    }
+
+
+    public ClausIE(LexicalizedParser lp, TokenizerFactory<CoreLabel> tokenizerFactory,
+            LexicalizedParserQuery lpq) {
+        this(new Options());
+        this.lp = lp;
+        this.tokenizerFactory = tokenizerFactory;
+        this.lpq = lpq;
+    }
+
+    // -- misc method
+    // ---------------------------------------------------------------------------
+
+    public Options getOptions() {
+        return options;
+    }
+
+    public void clear() {
+        semanticGraph = null;
+        depTree = null;
+        clauses.clear();
+        propositions.clear();
+    }
+
+    // -- parsing
+    // ---------------------------------------------------------------------------
+
+    /** Initializes the Stanford parser. */
+    public void initParser() {
+        lp = LexicalizedParser
+            .loadModel("edu/stanford/nlp/models/lexparser/englishPCFG.ser.gz");
+        tokenizerFactory = PTBTokenizer
+            .factory(new CoreLabelTokenFactory(), "");
+        lpq = lp.lexicalizedParserQuery();
+        morphology = new Morphology();
+    }
+
+
+    /** Clears and parses a new sentence. */
+    public void parse(String sentence) {
+        clear();
+        baseParse(sentence);
+        depTree = lpq.getBestParse();
+        semanticGraph = SemanticGraphFactory.generateUncollapsedDependencies(depTree);
+
+        for(IndexedWord iw: semanticGraph.vertexSet()) {
+            iw.setLemma(morphology.lemma(iw.word(), iw.tag()));
+        }
+    }
+
+    public void kparse(String sentence) {
+        clear();
+        baseParse(sentence);
+        trees =  lpq.getKBestPCFGParses(k);
+    }
+
+    private void baseParse(String sentence) {
+        List<CoreLabel> tokenizedSentence = tokenizerFactory.getTokenizer(
+            new StringReader(sentence)).tokenize();
+        lpq.parse(tokenizedSentence); // what about the confidence?
+    }
+
+    /** Returns the constituent tree for the sentence. */
+    public Tree getDepTree() {
+        return depTree;
+    }
+
+    /** Returns the dependency tree for the sentence. */
```

```
+    public SemanticGraph getSemanticGraph() {
+        return semanticGraph;
+    }
+
+    // -- clause detection
+    // ---------------------------------------------------------------------
+
+    /** Detects clauses in the sentence. */
+    public void detectClauses() {
+        ClauseDetector.detectClauses(this.options, this.semanticGraph, this.depTree, this.clauses);
+    }
+
+    /** Returns clauses in the sentence. */
+    public List<Clause> getClauses() {
+        return clauses;
+    }
+
+    // -- proposition generation
+    // ---------------------------------------------------------------------
+
+    /** Generates propositions from the clauses in the sentence. */
+    public void generatePropositions() {
+        propositions.clear();
+
+        // holds alternative options for each constituents (obtained by
+        // processing coordinated conjunctions and xcomps)
+        final List<List<Constituent>> constituents = new ArrayList<List<Constituent>>();
+
+        // which of the constituents are required?
+        final List<Flag> flags = new ArrayList<Flag>();
+        final List<Boolean> include = new ArrayList<Boolean>();
+
+        // holds all valid combination of constituents for which a proposition
+        // is to be generated
+        final List<List<Boolean>> includeConstituents = new ArrayList<List<Boolean>>();
+
+        // let's start
+        for (Clause clause : clauses) {
+            // process coordinating conjunctions
+            constituents.clear();
+            for (int i = 0; i < clause.getConstituents().size(); i++) {
+                // if(xcomp && clause.subject == i) continue; //An xcomp does
+                // not have an internal subject so should not be processed here
+                Constituent constituent = clause.getConstituents().get(i);
+                List<Constituent> alternatives;
+                if (!(xcomp && clause.getSubject() == i)
+                  && constituent instanceof IndexedConstituent
+                  // the processing of the xcomps is done in Default
+                  // proposition generator.
+                  // Otherwise we get duplicate propositions.
+                  && !clause.getXcomps().contains(i)
+                  && ((i == clause.getVerb() && options.processCcAllVerbs) || (i != clause.getVerb() && options.processCcNonVerbs)))
{
+                    alternatives = ProcessConjunctions.processCC(depTree, constituent, false, false, Integer.MAX_VALUE, null, null);
+                } else if (!(xcomp && clause.getSubject() == i)
+                      && clause.getXcomps().contains(i)) {
+                    alternatives = new ArrayList<Constituent>();
+                    ClausIE xclausIE = new ClausIE(options);
+                    xclausIE.semanticGraph = semanticGraph;
+                    xclausIE.depTree = depTree;
+                    xclausIE.xcomp = true;
+                    xclausIE.clauses = ((XcompConstituent) clause.getConstituents()
+                            .get(i)).getClauses();
+                    xclausIE.generatePropositions();
+                    for (Proposition p : xclausIE.propositions) {
+                        StringBuilder sb = new StringBuilder();
+                        String sep = "";
+                        for (int j = 0; j < p.constituents.size(); j++) {
+                            if (j == 0)    // to avoid including the subjecct, We
```

```
+                                    continue;  // could also generate the prop
+                                // without the subject
+                                sb.append(sep);
+                                sb.append(p.constituents.get(j));
+                                sep = " ";
+                            }
+                            alternatives.add(new TextConstituent(sb.toString(),
+                                                constituent.type));
+                        }
+                    } else {
+                        alternatives = new ArrayList<Constituent>(1);
+                        alternatives.add(constituent);
+                    }
+                    constituents.add(alternatives);
+                }
+
+                // create a list of all combinations of constituents for which a
+                // proposition should be generated
+                includeConstituents.clear();
+                flags.clear();
+                include.clear();
+                for (int i = 0; i < clause.getConstituents().size(); i++) {
+                    Flag flag = clause.getFlag(i, options);
+                    flags.add(flag);
+                    include.add(!flag.equals(Flag.IGNORE));
+                }
+                if (options.nary) {
+                    // we always include all constituents for n-ary ouput
+                    // (optional parts marked later)
+                    includeConstituents.add(include);
+                } else {
+                    // triple mode; determine which parts are required
+                    for (int i = 0; i < clause.getConstituents().size(); i++) {
+                        include.set(i, flags.get(i).equals(Flag.REQUIRED));
+                    }
+
+                    // create combinations of required/optional constituents
+                    new Runnable() {
+                        int noOptional;
+
+                        @Override
+                        public void run() {
+                            noOptional = 0;
+                            for (Flag f : flags) {
+                                if (f.equals(Flag.OPTIONAL))
+                                    noOptional++;
+                            }
+                            run(0, 0, new ArrayList<Boolean>());
+                        }
+
+                        private void run(int pos, int selected, List<Boolean> prefix) {
+                            if (pos >= include.size()) {
+                                if (selected >= Math.min(options.minOptionalArgs,
+                                                noOptional)
+                                   && selected <= options.maxOptionalArgs) {
+                                    includeConstituents.add(new ArrayList<Boolean>(
+                                                        prefix));
+                                }
+                                return;
+                            }
+                            prefix.add(true);
+                            if (include.get(pos)) {
+                                run(pos + 1, selected, prefix);
+                            } else {
+                                if (!flags.get(pos).equals(Flag.IGNORE)) {
+                                    run(pos + 1, selected + 1, prefix);
+                                }
+                                prefix.set(prefix.size() - 1, false);
+                                run(pos + 1, selected, prefix);
```

```
+                                }
+                                prefix.remove(prefix.size() - 1);
+                            }
+                        }.run();
+                    }
+
+                    // create a temporary clause for which to generate a proposition
+                    final Clause tempClause = clause.clone();
+
+                    // generate propositions
+                    new Runnable() {
+                        @Override
+                        public void run() {
+                            // select which constituents to include
+                            for (List<Boolean> include : includeConstituents) {
+                                // now select an alternative for each constituent
+                                selectConstituent(0, include);
+                            }
+                        }
+
+                        void selectConstituent(int i, List<Boolean> include) {
+                            if (i < constituents.size()) {
+                                if (include.get(i)) {
+                                    List<Constituent> alternatives = constituents
+                                        .get(i);
+                                    for (int j = 0; j < alternatives.size(); j++) {
+                                        tempClause.getConstituents().set(i,
+                                                alternatives.get(j));
+                                        selectConstituent(i + 1, include);
+                                    }
+                                } else {
+                                    selectConstituent(i + 1, include);
+                                }
+                            } else {
+                                // everything selected; generate
+                                propositionGenerator.generate(propositions, tempClause,
+                                        include);
+                            }
+                        }
+                    }.run();
+                }
+            }
+
+            public Collection<Proposition> getPropositions() {
+                if(!kparse)
+                    return propositions;
+                else
+                    return scoredPropositions.keySet();
+            }
+
+            // -- command-line interface
+            // ------------------------------------------------------------------
+
+            public static void main(String[] args) throws IOException {
+                OptionParser optionParser = new OptionParser();
+                optionParser
+                    .accepts("f",
+                            "input file (if absent, ClausIE reads from stdin)")
+                    .withRequiredArg().describedAs("file").ofType(String.class);
+                optionParser
+                    .accepts(
+                            "l",
+                            "if set, sentence identifier is read from input file (with lines of form: <id>\\t<sentence>)");
+                optionParser
+                    .accepts("o",
+                            "output file (if absent, ClausIE writes to stdout)")
+                    .withRequiredArg().describedAs("file").ofType(String.class);
+                optionParser.accepts("c", "configuration file").withRequiredArg()
+                    .describedAs("file").ofType(String.class);
```

```
+           optionParser.accepts("v", "verbose output");
+           optionParser.accepts("h", "print help");
+           optionParser.accepts("s", "print sentence");
+           optionParser.accepts("p", "print best parse confidence");
+           optionParser.accepts("k", "extract from k best parses (default 10)");
+           OptionSet options;
+           try {
+                options = optionParser.parse(args);
+           } catch (OptionException e) {
+                System.err.println(e.getMessage());
+                out.println("");
+                optionParser.printHelpOn(System.out);
+                return;
+           }
+           // help
+           if (options.has("h")) {
+                optionParser.printHelpOn(System.out);
+           }
+
+           // setup input and output
+           InputStream in = System.in;
+           OutputStream out = System.out;
+           if (options.has("f")) {
+                in = new FileInputStream((String) options.valueOf("f"));
+           }
+           if (options.has("o")) {
+                out = new FileOutputStream((String) options.valueOf("o"));
+           }
+
+           // is there an options file
+
+           // create a ClausIE instance and set options
+           final ClausIE clausIE;
+           if (options.has("c")) {
+                clausIE = new ClausIE(new Options((String) options.valueOf("c")));
+           } else {
+                clausIE = new ClausIE();
+           }
+           clausIE.initParser();
+           if (options.has("v")) {
+                clausIE.getOptions().print(out, "# ");
+           }
+
+           if (options.has("k")) {
+                if(options.valueOf("k") != null)
+                     clausIE.k = (Integer) options.valueOf("k");
+                clausIE.kparse = true;
+           }
+
+           // run
+           DataInput din = new DataInputStream(in);
+           PrintStream dout = new PrintStream(out);
+           int lineNo = 1;
+           for (String line = din.readLine(); line != null; line = din.readLine(), lineNo++) {
+                line = line.trim();
+                if (line.isEmpty() || line.startsWith("#"))
+                     continue;
+                int sentenceId = lineNo;
+                if (options.has("l")) {
+                     int tabIndex = line.indexOf('\t');
+                     sentenceId = Integer.parseInt(line.substring(0, tabIndex));
+                     line = line.substring(tabIndex + 1).trim();
+                }
+                if (options.has("v")) {
+                     dout.print("# Line ");
+                     dout.print(lineNo);
+                     if (options.has("l")) {
+                          dout.print(" (id ");
+                          dout.print(sentenceId);
```

```
+                         dout.print(")");
+                     }
+                     dout.print(": ");
+                     dout.print(line);
+                     dout.println();
+                 }
+
+                 if (!clausIE.kparse) {
+                     clausIE.parse(line);
+                     clausIE.detectClauses();
+                     try {
+                         clausIE.generatePropositions();
+                     } catch (java.lang.StackOverflowError e) {
+                         System.err.println("IGNORING KNOWN PROBLEM: StackOverflowError");
+                     }
+                 } else {
+                     clausIE.kparse(line);
+
+                     for(ScoredObject<Tree> tree: clausIE.trees) {
+                         clausIE.semanticGraph = SemanticGraphFactory
+                             .generateUncollapsedDependencies(tree.object());
+                         clausIE.depTree = tree.object();
+                         double score = Math.exp(tree.score());
+                         clausIE.detectClauses();
+                         clausIE.generatePropositions();
+                         //To store the best tree
+                         if(score > clausIE.bestScore) {
+                             clausIE.bestSemanticGraph = clausIE.semanticGraph;
+                             clausIE.bestDT = clausIE.depTree;
+                             clausIE.bestScore = score;
+                             clausIE.bestClauses = new ArrayList<Clause> (clausIE.clauses);
+                         }
+
+
+                         for (Proposition p : clausIE.propositions) {
+                             if(!clausIE.scoredPropositions.containsKey(p)) {
+                                 clausIE.scoredPropositions.put(p, score);
+                             } else {
+                                 clausIE.scoredPropositions.put(p,
+                                         clausIE.scoredPropositions.get(p) + score);
+                             }
+
+                         }
+                         clausIE.clear();
+                     }
+                 }
+
+                 clausIE.scoredPropositions = MapUtil.sortByValue(clausIE.scoredPropositions);
+
+                 if (options.has("v")) {
+                     if(clausIE.kparse) {
+                         clausIE.semanticGraph = clausIE.bestSemanticGraph;
+                     }
+                     dout.print("# Best Semantic graph: ");
+                     dout.println(clausIE.getSemanticGraph().toFormattedString()
+                             .replaceAll("\n", "\n#               ").trim());
+                 }
+
+                 if (options.has("v")) {
+                     if(clausIE.kparse) {
+                         clausIE.clauses = clausIE.bestClauses;
+                     }
+                     dout.print("#   Detected ");
+                     dout.print(clausIE.getClauses().size());
+                     dout.println(" clause(s).");
+                     for (Clause clause : clausIE.getClauses()) {
+                         dout.print("#   - ");
+                         dout.print(clause.toString(clausIE.options));
+                         dout.println();
```

```
+                       }
+                   }
+
+                   if (options.has("s")) {
+                       dout.print(line);
+                       dout.println();
+                   }
+
+                   for (Proposition p : clausIE.getPropositions()) {
+                       dout.print(sentenceId);
+                       for (String c : p.constituents) {
+                           // TODO: correct escaping
+                           dout.print("\t\"");
+                           dout.print(c);
+                           dout.print("\"");
+                       }
+                       if(clausIE.kparse) {
+                           dout.print("\t");
+                           dout.print(clausIE.scoredPropositions.get(p));
+                       }else if (options.has("p")) {
+                           dout.print("\t");
+                           dout.print(clausIE.lpq.getPCFGScore());
+                       }
+                       dout.println();
+                   }
+               }
+
+
+
+                       // shutdown
+                       if (options.has("f")) {
+                                   in.close();
+                       }
+                       if (options.has("o")) {
+                                   out.close();
+                       }
+            }
+}
diff --git a/src/clausie/Clause.java b/src/clausie/Clause.java
new file mode 100644
index 0000000..bc56a59
--- /dev/null
+++ b/src/clausie/Clause.java
@@ -0,0 +1,739 @@
+package clausie;
+
+import java.util.ArrayList;
+import java.util.List;
+import java.util.LinkedHashMap;
+import java.util.HashMap;
+import java.util.Map;
+
+import clausie.Constituent.Flag;
+import edu.stanford.nlp.ling.IndexedWord;
+import edu.stanford.nlp.semgraph.SemanticGraph;
+import edu.stanford.nlp.trees.Tree;
+import edu.stanford.nlp.ling.CoreAnnotations;
+import edu.stanford.nlp.ling.CoreLabel;
+import edu.stanford.nlp.trees.EnglishGrammaticalRelations;
+/**
+ * A clause is a basic unit of a sentence. In ClausIE, a clause consists of a
+ * set of constituents (at least a subject and a verb) and a type.
+ */
+public class Clause {
+           // -- Type definition
+           // ------------------------------------------------------------------------
+
+           public static Integer IDs = null;
+
```

```
+	public int clauseID;
+
+	/** Clause types */
+	public enum Type {
+		SV, SVC, SVA, SVO, SVOO, SVOC, SVOA, EXISTENTIAL, UNKNOWN;
+	};
+
+	// -- member variables
+	// ---------------------------------------------------------------------
+
+	protected SemanticGraph semanticGraph;
+
+	protected Tree constTree;
+
+	/** Constituents of this clause */
+	public List<Constituent> constituents = new ArrayList<Constituent>();
+
+	/** Type of this clause */
+	protected Type type = Type.UNKNOWN;
+
+	/** Position of subject in {@link #constituents} */
+	protected int subject = -1;
+
+	/** Position of verb in {@link #constituents} */
+	protected int verb = -1;
+
+	protected boolean cop;
+
+	protected boolean passive;
+
+	private Boolean isNegated = null;
+
+	public List<IndexedWord> clauseMembers;
+
+
+	// They are lists because some times the parsers (probably an error)
+	// generates more than one constituent of each type
+	// e.g., more than one dobj produced by parser for
+	// "The man who I told the fact is dead."
+	/** Position(s) of direct object(s) in {@link #constituents}. */
+	protected List<Integer> dobjects = new ArrayList<Integer>();
+
+	/** Position(s) of indirect object in {@link #constituents} */
+	protected List<Integer> iobjects = new ArrayList<Integer>();
+
+	/** Position of complement in {@link #constituents} (for SVC / SVOC) */
+	protected int complement = -1;
+
+	/** Position(s) of xcomps in {@link #constituents} */
+	protected List<Integer> xcomps = new ArrayList<Integer>();
+
+	/** Position(s) of ccomps in {@link #constituents} */
+	protected List<Integer> ccomps = new ArrayList<Integer>();
+
+	/** Position(s) of acomps in {@link #constituents} */
+	protected List<Integer> acomps = new ArrayList<Integer>();
+
+	/** Position(s) of adverbials in {@link #constituents} */
+	protected List<Integer> adverbials = new ArrayList<Integer>();
+
+	/** Non identified dependencies */
+	protected List<Integer> deps = new ArrayList<Integer>();
+
+	/** If a relative pronoun refers to an adverbial */
+	protected boolean relativeAdverbial = false;
+
+	/**
+	 * Parent clause of this clause, if any. For example, in
+	 * "He said this is true." the clause "this / is / true" has parent
```

```
+     * "he / said / this is true".
+     */
+    protected Clause parentClause = null;
+
+    /** Agent (for passive voice). Currently unused. */
+    private IndexedWord agent;
+
+    /** Root of the clause. */
+    protected IndexedWord root;
+
+
+    // -- construction
+    // ---------------------------------------------------------------------------
+
+    // make package private
+    public Clause() {
+        if(IDs == null)
+            IDs = 0;
+        clauseID = IDs;
+        IDs++;
+    };
+
+    @Override
+    public Clause clone() {
+        Clause clause = new Clause();
+        clause.setConstituents(new ArrayList<Constituent>(getConstituents()));
+        clause.setType(type);
+        clause.setSubject(subject);
+        clause.setVerb(verb);
+        clause.setRoot(root);
+        clause.setDobjects(new ArrayList<Integer>(getDobjects()));
+        clause.setIobjects(new ArrayList<Integer>(getIobjects()));
+        clause.setComplement(complement);
+        clause.setXcomps(new ArrayList<Integer>(getXcomps()));
+        clause.setCcomps(new ArrayList<Integer>(getCcomps()));
+        clause.setAcomps(new ArrayList<Integer>(getAcomps()));
+        clause.setAdverbials(new ArrayList<Integer>(getAdverbials()));
+        clause.setRelativeAdverbial(relativeAdverbial);
+        clause.setAgent(agent);
+        clause.setParentClause(parentClause);
+        return clause;
+    }
+
+    // -- methods
+    // ---------------------------------------------------------------------------
+
+    /** Determines the type of this clause, if still unknown. */
+    void detectType(Options options) {
+        if (getType() != Type.UNKNOWN)
+            return;
+
+        // count the total number of complements (dobj, ccomp, xcomp)
+        int noComplements = noComplements();
+
+        // sometimes the parsers gives ccomp and xcomp instead of direct objects
+        // e.g., "He is expected to tell the truth."
+        IndexedWord root = ((IndexedConstituent) getConstituents().get(getVerb()))
+            .getRoot();
+        boolean hasDirectObject = getDobjects().size() > 0
+            || (getComplement() < 0 && noComplements > 0 && !options.isCop(root));
+        boolean hasIndirectObject = !getIobjects().isEmpty();
+
+        // Q1: Object?
+        if (hasDirectObject || hasIndirectObject) {
+            // Q7: dir. and indir. object?
+            if (noComplements > 0 && hasIndirectObject) {
+                setType(Type.SVOO);
+                return;
+            }
```

```
+
+                // Q8: Complement?
+                if (noComplements > 1) {
+                    setType(Type.SVOC);
+                    return;
+                }
+
+                // Q9: Candidate adverbial and direct objects?
+                if (!(hasCandidateAdverbial() && hasDirectObject)) {
+                    setType(Type.SVO);
+                    return;
+                }
+
+                // Q10: Potentially complex transitive?
+                if (options.isComTran(root)) {
+                    setType(Type.SVOA);
+                    return;
+                }
+
+                // Q11: Conservative?
+                if (options.conservativeSVOA) {
+                    setType(Type.SVOA);
+                    return;
+                } else {
+                    setType(Type.SVO);
+                    return;
+                }
+            } else {
+                // Q2: Complement?
+                // not sure about acomp, can a copular be transitive?
+                if (getComplement() >= 0 || noComplements > 0 && options.isCop(root)
+                   || !getAcomps().isEmpty()) {
+                    setType(Type.SVC);
+                    return;
+                }
+
+                // Q3: Candidate adverbial
+                if (!hasCandidateAdverbial()) {
+                    setType(Type.SV);
+                    return;
+                }
+
+                // Q4: Known non ext. copuular
+                if (options.isNotExtCop(root)) {
+                    setType(Type.SV);
+                    return;
+                }
+
+                // Q5: Known ext. copular
+                if (options.isExtCop(root)) {
+                    setType(Type.SVA);
+                    return;
+                }
+
+                // Q6: Conservative
+                if (options.conservativeSVA) {
+                    setType(Type.SVA);
+                    return;
+                } else {
+                    setType(Type.SV);
+                    return;
+                }
+            }
+        }
+
+        /**
+         * Checks whether this clause has a candidate adverbial, i.e., an adverbial
+         * that can potentially be obligatory.
+         */
```

```
+  public String getCandidateAdverbial() {
+          if (getAdverbials().isEmpty())
+              return null;
+          if (isRelativeAdverbial())
+            return "FIXME";
+
+          // is there an adverbial that occurs after the verb?
+          IndexedWord a = ((IndexedConstituent) getConstituents().get(getAdverbials().get(getAdverbials().size() - 1))).getRoot();
+          IndexedWord b = ((IndexedConstituent) getConstituents().get(getVerb())).getRoot();
+          System.out.format("ADVERBS: a: %s\tb: %s\tas: %s\n", a, b, getAdverbials().size());
+          String str = String.format("%s|%s", a.backingLabel().get(CoreAnnotations.TextAnnotation.class),
a.backingLabel().get(CoreAnnotations.PartOfSpeechAnnotation.class));
+          if (a.index() > b.index())
+            return str;
+          return null;
+      }
+
+      public boolean hasCandidateAdverbial() {
+          if (getAdverbials().isEmpty())
+              return false;
+          if (isRelativeAdverbial())
+              return true;
+
+          // is there an adverbial that occurs after the verb?
+          if (((IndexedConstituent) getConstituents().get(getAdverbials().get(getAdverbials()
+                                                  .size() - 1))).getRoot().index() > ((IndexedConstituent) getConstituents()
+                                                                  .get(getVerb())).getRoot().index())
+              return true;
+          return false;
+      }
+
+      /**
+       * Determines the total number of complements (includes direct objects,
+       * subject complements, etc.) present in this clause.
+       */
+      public int noComplements() {
+        int c = (getComplement() < 0 ? 0 : 1);
+        int d = getDobjects().size();
+        int x = getXcomps().size();
+        int cc= getCcomps().size();
+        System.out.format("d: %s\tc: %s\tx: %s\tcc: %s\n", d, c, x, cc);
+
+          return getDobjects().size() + (getComplement() < 0 ? 0 : 1) + getXcomps().size()
+              + getCcomps().size();
+      }
+
+      @Override
+      public String toString() {
+          return toString(null);
+      }
+
+      public String toString(Options options) {
+          Clause clause = this;
+          StringBuffer s = new StringBuffer();
+          s.append(clause.getType().name());
+          s.append(" (");
+          String sep = "";
+          for (int index = 0; index < getConstituents().size(); index++) {
+              Constituent constituent = getConstituents().get(index);
+              s.append(sep);
+              sep = ", ";
+              switch (constituent.getType()) {
+              case ACOMP:
+                  s.append("ACOMP");
+                  break;
+              case ADVERBIAL:
+                  s.append("A");
+                  if (options != null) {
+                      switch (getFlag(index, options)) {
```

```
+                           case IGNORE:
+                               s.append("-");
+                               break;
+                           case OPTIONAL:
+                               s.append("?");
+                               break;
+                           case REQUIRED:
+                               s.append("!");
+                               break;
+                           }
+                       }
+                       break;
+                   case CCOMP:
+                       s.append("CCOMP");
+                       break;
+                   case COMPLEMENT:
+                       s.append("C");
+                       break;
+                   case DOBJ:
+                       s.append("O");
+                       break;
+                   case IOBJ:
+                       s.append("IO");
+                       break;
+                   case SUBJECT:
+                       s.append("S");
+                       break;
+                   case UNKOWN:
+                       s.append("?");
+                       break;
+                   case VERB:
+                       s.append("V");
+                       break;
+                   case XCOMP:
+                       s.append("XCOMP");
+                       break;
+                   }
+                   s.append(": ");
+                   if (!(constituent instanceof IndexedConstituent)) {
+                       s.append("\"");
+                   }
+                   s.append(constituent.rootString());
+                   if (constituent instanceof IndexedConstituent) {
+                       s.append("@");
+                       s.append(((IndexedConstituent) constituent).getRoot().index());
+                   } else {
+                       s.append("\"");
+                   }
+               }
+               s.append(")");
+               return s.toString();
+           }
+
+
+           public Map<String,String> svoMap(Options options) {
+               Clause clause = this;
+               Map<String,String> m = new LinkedHashMap<String,String>();
+               for (int index = 0; index < getConstituents().size(); index++) {
+                   Constituent constituent = getConstituents().get(index);
+                   String root = constituent.rootString();
+                   Constituent.Type cType = constituent.getType();
+                   switch (cType) {
+                   case ADVERBIAL:
+                       if (options != null) {
+                           Constituent.Flag flag = getFlag(index, options);
+                           switch (flag) {
+                           // case IGNORE:
+                           //      s.append("-");
+                           //      break;
```

```
+                           // case OPTIONAL:
+                           //       s.append("?");
+                           //       break;
+                           case REQUIRED:
+                                m.put(cType.toString() + flag.toString(), root);
+                                break;
+                           }
+                      }
+                      break;
+                 case VERB:
+                      if (constituent instanceof IndexedConstituent) {
+                           IndexedConstituent ic = (IndexedConstituent) constituent;
+                           IndexedWord iw = ic.getRoot();
+                           CoreLabel label = ic.getRoot().backingLabel();
+                           if (root == "") {
+                            root = label.get(CoreAnnotations.OriginalTextAnnotation.class);
+                           }
+                           m.put(cType.toString(), root);
+                           String dfFile = label.get(CoreAnnotations.DFTypeAnnotation.class);
+                           if (dfFile != null) {
+                            m.put("TUAfile", dfFile);
+                           }
+                           String dfType = label.get(CoreAnnotations.DFTypeIDAnnotation.class);
+                           if (dfType != null) {
+                            String[] parts = dfType.split("\\|");
+                            m.put("TUAid", dfType);
+                            m.put("TUAtype", parts[0]);
+                            m.put("TUAnum", parts[1]);
+                            m.put("TUAcolor", parts[2]);
+                           }
+                           String metadata = label.get(CoreAnnotations.DocIDAnnotation.class);
+                           if (metadata != null ) {
+                            String[] parts = metadata.split("\\|");
+                            m.put("metadata", metadata);
+                            m.put("city", parts[0]);
+                            m.put("article_date", parts[1]);
+                            m.put("coders", parts[2]);
+                            m.put("srcfile", parts[3]);
+                           }
+
+                           // String docType = label.get(CoreAnnotations.DocTypeAnnotation.class); if (docType != null) m.put("city",
docType);
+                           // String docDate = label.get(CoreAnnotations.DocDateAnnotation.class); if (docDate != null) m.put("date",
docDate);
+                           // String docSourceType = label.get(CoreAnnotations.DocSourceTypeAnnotation.class); if (docSourceType != null)
m.put("annotators", docSourceType);
+                           if (label.lemma() != null) m.put("Lemma", label.lemma());
+                           if (clause.hasCandidateAdverbial()) {
+                            // IndexedWord child = semanticGraph.getChildWithReln(iw, EnglishGrammaticalRelations.AUX_MODIFIER);
+                            // m.put("hasCandidateAdverbial", child.toString());
+                            m.put("hasCandidateAdverbial", clause.getCandidateAdverbial());
+                           }
+
+                           if (clause.isNegated(options)) {
+                                // m.put("NEGATED", iw.toString());
+                                IndexedWord child = semanticGraph.getChildWithReln(iw,
EnglishGrammaticalRelations.NEGATION_MODIFIER);
+                                m.put("NEGATED", child.toString());
+                           }
+                      }
+                      break;
+                 default:
+                      m.put(cType.toString(), root);
+                      break;
+                 }
+            }
+            if (clause.getPassive()) {
+                 String s = m.get("S");
+                 String o = m.get("O");
```

```
+                if (o != null && s != null) {
+                    m.put("NSUBJPASS", "swap");
+                    m.put("S", o);
+                    m.put("O", s);
+                } else if (s != null) {
+                    m.put("NSUBJPASS", "move");
+                    m.put("O", s);
+                    m.remove("S");
+                } else {
+                    m.put("NSUBJPASS", "unknown");
+                }
+            }
+            return m;
+        }
+
+        /**
+         * Determines the flag of the adverbial at position {@code index} in
+         * {@link #adverbials}, i.e., whether the adverbial is required, optional,
+         * or to be ignored.
+         */
+        public Flag getFlag(int index, Options options) {
+
+            boolean first = true;
+            for (int i : getAdverbials()) {
+                if (i == index && isIgnoredAdverbial(i, options))
+                    return Flag.IGNORE;
+                else if (i == index && isIncludedAdverbial(i, options))
+                    return Flag.REQUIRED;
+                int adv = ((IndexedConstituent) getConstituents().get(i)).getRoot()
+                        .index();
+                if (getConstituents().get(getVerb()) instanceof IndexedConstituent
+                   && adv < ((IndexedConstituent) getConstituents().get(getVerb()))
+                   .getRoot().index() && !isRelativeAdverbial()) {
+                    if (i == index) {
+                        return Flag.OPTIONAL;
+                    }
+                } else {
+                    if (i == index) {
+                        if (!first)
+                            return Flag.OPTIONAL;
+                        return !(Type.SVA.equals(getType()) || Type.SVOA.equals(getType())) ? Flag.OPTIONAL
+                                : Flag.REQUIRED;
+                    }
+                    first = false;
+                }
+            }
+            return Flag.REQUIRED;
+        }
+
+        /**
+         * Checks whether the adverbial at position {@code index} in
+         * {@link #adverbials} is to be ignored by ClausIE.
+         */
+        private boolean isIgnoredAdverbial(int index, Options options) {
+            Constituent constituent = getConstituents().get(index);
+            String s;
+            if (constituent instanceof IndexedConstituent) {
+                IndexedConstituent indexedConstituent = (IndexedConstituent) constituent;
+                IndexedWord root = indexedConstituent.getRoot();
+                if (indexedConstituent.getSemanticGraph().hasChildren(root)) {
+                    // ||IndexedConstituent.sentSemanticGraph.getNodeByIndexSafe(root.index()
+                    // + 1) != null
+                    // &&
+                    // IndexedConstituent.sentSemanticGraph.getNodeByIndexSafe(root.index()
+                    // + 1).tag().charAt(0) == 'J') { //do not ignore if it modifies
+                    // an adjective. Adverbs can modify verbs or adjective no reason
+                    // to ignore them when they refer to adjectives (at lest in
+                    // triples). This is important in the case of adjectival
+                    // complements
```

```
+                    return false;
+                }
+                s = root.lemma();
+            } else {
+                s = constituent.rootString();
+            }
+
+            if (options.dictAdverbsIgnore.contains(s)
+                || (options.processCcNonVerbs && options.dictAdverbsConj
+                .contains(s)))
+                return true;
+            else
+                return false;
+    }
+
+    /**
+     * Checks whether the adverbial at position {@code index} in
+     * {@link #adverbials} is required to be output by ClausIE (e.g., adverbials
+     * indicating negation, such as "hardly").
+     */
+    private boolean isIncludedAdverbial(int index, Options options) {
+            Constituent constituent = getConstituents().get(index);
+            String s;
+            if (constituent instanceof IndexedConstituent) {
+                IndexedConstituent indexedConstituent = (IndexedConstituent) constituent;
+                IndexedWord root = indexedConstituent.getRoot();
+                if (indexedConstituent.getSemanticGraph().hasChildren(root)) {
+                    return false;
+                }
+                s = root.lemma();
+            } else {
+                s = constituent.rootString();
+            }
+            return options.dictAdverbsInclude.contains(s);
+    }
+
+    public SemanticGraph getSemanticGraph () {
+            return semanticGraph;
+    }
+
+    public void setSemanticGraph (SemanticGraph semanticGraph) {
+            this.semanticGraph = semanticGraph;
+    }
+
+    public int getVerb() {
+            return verb;
+    }
+
+    public void setVerb(int verb) {
+            this.verb = verb;
+    }
+
+    public void setRoot(IndexedWord root) {
+            this.root = root;
+    }
+
+    public int getComplement() {
+            return complement;
+    }
+
+    public void setComplement(int complement) {
+            this.complement = complement;
+    }
+
+    public List<Constituent> getConstituents() {
+            return constituents;
+    }
+
+    public void setConstituents(List<Constituent> constituents) {
```

```
+            this.constituents = constituents;
+        }
+
+        public int getSubject() {
+            return subject;
+        }
+
+        public void setSubject(int subject) {
+            this.subject = subject;
+        }
+
+        public List<Integer> getIobjects() {
+            return iobjects;
+        }
+
+        public void setIobjects(List<Integer> iobjects) {
+            this.iobjects = iobjects;
+        }
+
+        public List<Integer> getDobjects() {
+            return dobjects;
+        }
+
+        public void setDobjects(List<Integer> dobjects) {
+            this.dobjects = dobjects;
+        }
+
+        public List<Integer> getCcomps() {
+            return ccomps;
+        }
+
+        public List<Integer> getDeps() {
+            return deps;
+        }
+
+        public void setCcomps(List<Integer> ccomps) {
+            this.ccomps = ccomps;
+        }
+
+        public List<Integer> getXcomps() {
+            return xcomps;
+        }
+
+        public void setXcomps(List<Integer> xcomps) {
+            this.xcomps = xcomps;
+        }
+
+        public List<Integer> getAcomps() {
+            return acomps;
+        }
+
+        public void setAcomps(List<Integer> acomps) {
+            this.acomps = acomps;
+        }
+
+        public List<Integer> getAdverbials() {
+            return adverbials;
+        }
+
+        public void setAdverbials(List<Integer> adverbials) {
+            this.adverbials = adverbials;
+        }
+
+        public void setDeps(ArrayList<Integer> deps) {
+            this.deps = deps;
+        }
+
+
+        public Type getType() {
```

```
+           return type;
+       }
+
+       public void setType(Type type) {
+           this.type = type;
+       }
+
+       public boolean getCop() {
+           return cop;
+       }
+
+       public boolean getPassive() {
+           return passive;
+       }
+
+       public IndexedWord getRoot() {
+           return root;
+       }
+
+       public void setCop(boolean cop) {
+           this.cop = cop;
+       }
+
+       public Clause getParentClause() {
+           return parentClause;
+       }
+
+       public void setParentClause(Clause parentClause) {
+           this.parentClause = parentClause;
+       }
+
+       public boolean isRelativeAdverbial() {
+           return relativeAdverbial;
+       }
+
+       protected void setRelativeAdverbial(boolean relativeAdverbial) {
+           this.relativeAdverbial = relativeAdverbial;
+       }
+
+       public void setPassive(boolean b) {
+           passive = b;
+
+       }
+
+       public void setAgent(IndexedWord agent) {
+           this.agent = agent;
+
+       }
+
+       public IndexedWord getAgent() {
+           return agent;
+       }
+
+       public Tree getTree() {
+           return constTree;
+       }
+
+       public void setTree(Tree constTree) {
+           this.constTree = constTree;
+       }
+
+       public List<IndexedWord> getClauseMembers(List<IndexedWord> clauseRoots) {
+           List<IndexedWord> result = new ArrayList<IndexedWord>();
+           result.add(root);
+           getClauseMembers(semanticGraph.getChildList(root), clauseRoots, result);
+           return result;
+       }
+
+       private void getClauseMembers(List<IndexedWord> childList,
```

```
+                       List<IndexedWord> clauseRoots, List<IndexedWord> result) {
+           for(IndexedWord child: childList) {
+                if(clauseRoots.contains(child) || child.tag().charAt(0) == 'W' || result.contains(child))
+                     continue;
+                result.add(child);
+                getClauseMembers(semanticGraph.getChildList(child), clauseRoots, result);
+           }
+      }
+
+       public boolean isNegated(Options options) {
+                if(isNegated == null) {
+                        if(semanticGraph.isNegatedVertex(root)) {
+                                isNegated = true;
+                        }else {
+                                for(int adverbial: adverbials) {
+                                        IndexedConstituent c = (IndexedConstituent) constituents.get(adverbial);
+                                        if(options.dictAdverbsInclude.contains(c.getRoot().lemma())) {
+                                                isNegated = true;
+                                                break;
+                                        }
+                                }
+                                if(isNegated == null)
+                                        isNegated = false;
+                        }
+                }
+
+                return isNegated;
+       }
+}
diff --git a/src/clausie/ClauseDetector.java b/src/clausie/ClauseDetector.java
new file mode 100644
index 0000000..3a756a1
--- /dev/null
+++ b/src/clausie/ClauseDetector.java
@@ -0,0 +1,857 @@
+package clausie;
+
+import java.util.ArrayList;
+import java.util.Collections;
+import java.util.HashSet;
+import java.util.List;
+import java.util.Set;
+import java.util.TreeSet;
+
+import edu.stanford.nlp.ling.IndexedWord;
+import edu.stanford.nlp.semgraph.SemanticGraph;
+import edu.stanford.nlp.semgraph.SemanticGraphEdge;
+import edu.stanford.nlp.semgraph.SemanticGraphFactory;
+import edu.stanford.nlp.trees.EnglishGrammaticalRelations;
+import edu.stanford.nlp.trees.GrammaticalRelation;
+import edu.stanford.nlp.trees.Tree;
+
+/**{@link ClauseDetector} contains the methods dealing with the detection of clauses.
+ * After the detection is performed a set of {@link Clause} is created.
+ *
+ * {@code detectClauses} first detects the type of clause to be generated based on syntactic relations
+ * and once a clause is detected a given method is used to create a {@link Clause}.
+ *
+ * @date $LastChangedDate: 2013-12-02 15:45:41 +0100 (Mon, 02 Dec 2013) $
+ * @version $LastChangedRevision: 1182 $ */
+public class ClauseDetector {
+
+       /** Set of dependency relations that do not belong to a complement */
+       protected static final Set<GrammaticalRelation> EXCLUDE_RELATIONS_COMPLEMENT;
+       static {
+            HashSet<GrammaticalRelation> temp = new HashSet<GrammaticalRelation>();
+            temp.add(EnglishGrammaticalRelations.AUX_MODIFIER);
+            temp.add(EnglishGrammaticalRelations.AUX_PASSIVE_MODIFIER);
+            temp.add(EnglishGrammaticalRelations.SUBJECT);
```

```
+            temp.add(EnglishGrammaticalRelations.COPULA);
+            temp.add(EnglishGrammaticalRelations.ADVERBIAL_MODIFIER);
+            EXCLUDE_RELATIONS_COMPLEMENT = Collections.unmodifiableSet(temp);
+        }
+
+        /** Set of dependency relations that belong to the verb */
+        protected static final Set<GrammaticalRelation> INCLUDE_RELATIONS_VERB;
+        static {
+            HashSet<GrammaticalRelation> temp = new HashSet<GrammaticalRelation>();
+            temp.add(EnglishGrammaticalRelations.AUX_MODIFIER);
+            temp.add(EnglishGrammaticalRelations.AUX_PASSIVE_MODIFIER);
+            temp.add(EnglishGrammaticalRelations.NEGATION_MODIFIER);
+            INCLUDE_RELATIONS_VERB = Collections.unmodifiableSet(temp);
+        }
+
+        private ClauseDetector() {
+        };
+
+        /** Detects clauses in the input sentence */
+        public static void detectClauses(Options options, SemanticGraph semanticGraph, Tree depTree, List<Clause> clauses) {
+            IndexedConstituent.sentSemanticGraph = semanticGraph;
+            List<IndexedWord> roots = new ArrayList<IndexedWord>();
+            boolean includeXcomps = true;
+            for (SemanticGraphEdge edge : semanticGraph.edgeIterable()) {
+                // check whether the edge identifies a clause
+                if (DpUtils.isAnySubj(edge)) {
+                    // clauses with a subject
+                    IndexedWord subject = edge.getDependent();
+                    IndexedWord root = edge.getGovernor();
+                    addNsubjClause(semanticGraph, depTree, options, roots, clauses, subject, root, false, includeXcomps); //CHECK
+                } else if (options.processAppositions && DpUtils.isAppos(edge)) {
+                    // clauses for appositions
+                    IndexedWord subject = edge.getGovernor();
+                    IndexedWord object = edge.getDependent();
+                    addApposClause(depTree, semanticGraph, options, clauses, subject, object);
+                    roots.add(null);
+                } else if (options.processPossessives && DpUtils.isPoss(edge)) {
+                    // clauses for possessives
+                    IndexedWord subject = edge.getDependent();
+                    IndexedWord object = edge.getGovernor();
+                    addPossessiveClause(depTree, semanticGraph, options, clauses, subject, object);
+                    roots.add(null);
+                } else if (options.processPartmods && DpUtils.isPartMod(edge)) {
+                    // clauses for participial modifiers
+                    IndexedWord subject = edge.getGovernor();
+                    IndexedWord object = edge.getDependent();
+                    addPartmodClause(semanticGraph, depTree, options, clauses, subject, object, roots, includeXcomps);
+                }
+            }
+
+            // postprocess clauses
+            // TODO
+            for (int i = 0; i < clauses.size(); i++) {
+                Clause clause = clauses.get(i);
+
+                // set parents (slow and inefficient for now)
+                IndexedWord root = roots.get(i);
+                if (root != null) {
+                    int index = ancestorOf(semanticGraph, root, roots); // recursion needed to
+                    // deal
+                    // with xcomp; more stable
+                    if (index >= 0) {
+                        // System.out.println("Clause " + clause.toString() + " has parent " +
+                        // clauses.get(index).toString());
+                        clause.setParentClause(clauses.get(index));
+                    }
+                }
+
+                // exclude vertexes (each constituent needs to excludes vertexes of the other
```

```
+                // constituents)
+                excludeVertexes(clause);
+            }
+        }
+
+
+        /** Detects clauses in the input sentence
+         * @return */
+        public static List<Clause> detectClauses(Options options, SemanticGraph semanticGraph, Tree depTree, List<IndexedWord> roots,
boolean includeXcomps) {
+            List<Clause> clauses = new ArrayList<Clause>();
+            for (SemanticGraphEdge edge : semanticGraph.edgeIterable()) {
+                // check whether the edge identifies a clause
+                if (DpUtils.isAnySubj(edge)) {
+                    // clauses with a subject
+                    IndexedWord subject = edge.getDependent();
+                    IndexedWord root = edge.getGovernor();
+                    addNsubjClause(semanticGraph, depTree, options, roots, clauses, subject, root, false, includeXcomps);
+                } else if (options.processAppositions && DpUtils.isAppos(edge)) {
+                    // clauses for appositions
+                    IndexedWord subject = edge.getGovernor();
+                    IndexedWord object = edge.getDependent();
+                    addApposClause(depTree, semanticGraph, options, clauses, subject, object);
+                    roots.add(null);
+                } else if (options.processPossessives && DpUtils.isPoss(edge)) {
+                    // clauses for possessives
+                    IndexedWord subject = edge.getDependent();
+                    IndexedWord object = edge.getGovernor();
+                    addPossessiveClause(depTree, semanticGraph, options, clauses, subject, object);
+                    roots.add(null);
+                }// else if (options.processPartmods && DpUtils.isPartMod(edge)) {
+                // clauses for participial modifiers
+                //   IndexedWord subject = edge.getGovernor();
+                //   IndexedWord object = edge.getDependent();
+                //   addPartmodClause(semanticGraph, depTree, options, clauses, subject, object, roots, includeXcomps);
+                // }
+            }
+
+            // postprocess clauses
+            // TODO
+            for (int i = 0; i < clauses.size(); i++) {
+                Clause clause = clauses.get(i);
+
+                // set parents (slow and inefficient for now)
+                IndexedWord root = roots.get(i);
+                if (root != null) {
+                    int index = ancestorOf(semanticGraph, root, roots); // recursion needed to
+                    // deal
+                    // with xcomp; more stable
+                    if (index >= 0) {
+                        // System.out.println("Clause " + clause.toString() + " has parent " +
+                        // clauses.get(index).toString());
+                        clause.setParentClause(clauses.get(index));
+                    }
+                }
+
+                // exclude vertexes (each constituent needs to excludes vertexes of the other
+                // constituents)
+                excludeVertexes(clause);
+            }
+            return clauses;
+        }
+
+        /** Detects clauses in the input sentence for a given subject
+         * @return */
+        public static List<Clause> detectClauses(Options options, SemanticGraph semanticGraph, Tree depTree,
+                                                 List<IndexedWord> roots, boolean includeXcomps, IndexedWord subject, IndexedWord root) {
+            List<Clause> clauses = new ArrayList<Clause>();
+            addNsubjClause(semanticGraph, depTree, options, roots, clauses, subject, root, false, includeXcomps);
```

```
+
+           for (int i = 0; i < clauses.size(); i++) {
+                Clause clause = clauses.get(i);
+
+                // set parents (slow and inefficient for now)
+                IndexedWord rootN = roots.get(i);
+                if (root != null) {
+                     int index = ancestorOf(semanticGraph, rootN, roots); // recursion needed to
+                     // deal
+                     // with xcomp; more stable
+                     if (index >= 0) {
+                          // System.out.println("Clause " + clause.toString() + " has parent " +
+                          // clauses.get(index).toString());
+                          clause.setParentClause(clauses.get(index));
+                     }
+                }
+
+                // exclude vertexes (each constituent needs to excludes vertexes of the other
+                // constituents)
+                excludeVertexes(clause);
+           }
+           return clauses;
+      }
+
+      /** Adds in the exclude vertex of a clause the head of the rest of the clauses */
+      public static void excludeVertexes(Clause clause) {
+
+           for (int j = 0; j < clause.getConstituents().size(); j++) {
+                if (!(clause.getConstituents().get(j) instanceof IndexedConstituent))
+                     continue;
+                IndexedConstituent constituent = (IndexedConstituent) clause.getConstituents().get(j);
+
+                for (int k = 0; k < clause.getConstituents().size(); k++) {
+                     if (k == j || !(clause.getConstituents().get(k) instanceof IndexedConstituent))
+                          continue;
+                     IndexedConstituent other = (IndexedConstituent) clause.getConstituents().get(k);
+
+                     constituent.getExcludedVertexes().add(other.getRoot());
+                     constituent.getExcludedVertexes().addAll(other.getAdditionalVertexes());
+                }
+           }
+
+      }
+
+      /** TODO */
+      private static int ancestorOf(SemanticGraph semanticGraph, IndexedWord node,
+                                    List<IndexedWord> ancestors) {
+           for (SemanticGraphEdge e : semanticGraph.getIncomingEdgesSorted(node)) {
+                int index = ancestors.indexOf(node);
+                if (index >= 0)
+                     return index;
+                index = ancestorOf(semanticGraph, e.getGovernor(), ancestors);
+                if (index >= 0)
+                     return index;
+           }
+           return -1;
+      }
+
+      /** Selects constituents of a clause for clauses with internal subject or coming from a participial modifier
+       * @param roots The list of roots of the clauses in the sentence
+       * @param clauses The list of clauses in the sentence
+       * @param subject The subject of the clause
+       * @param clauseRoot The root of the clause, either a verb or a complement
+       * @param partmod Indicates if the clause is generated from a partmod relation*/
+      private static void addNsubjClause(SemanticGraph oSemanticGraph, Tree depTree, Options options, List<IndexedWord> roots,
+                                         List<Clause> clauses, IndexedWord subject, IndexedWord clauseRoot, boolean partmod, boolean includeXcomps)
{
+           //SemanticGraph semanticGraph = new SemanticGraph(oSemanticGraph);
+           SemanticGraph semanticGraph = SemanticGraphFactory.duplicateKeepNodes(oSemanticGraph);
```

```
+            List<SemanticGraphEdge> toRemove = new ArrayList<SemanticGraphEdge>();
+            //to store the heads of the clauses according to the CCs options
+            List<IndexedWord> ccs = ProcessConjunctions.getIndexedWordsConj(semanticGraph,
+                                                depTree, clauseRoot, EnglishGrammaticalRelations.CONJUNCT, toRemove,
+                                                options);
+            for (SemanticGraphEdge edge : toRemove)
+                semanticGraph.removeEdge(edge);
+
+            //A new clause is generated for each clause head
+            for (int i = 0; i < ccs.size(); i++) {
+                IndexedWord root = ccs.get(i);
+                List<SemanticGraphEdge> outgoingEdges = semanticGraph.getOutEdgesSorted(root);
+                List<SemanticGraphEdge> incomingEdges = semanticGraph.getIncomingEdgesSorted(root);
+
+                SemanticGraphEdge cop = DpUtils.findFirstOfRelation(outgoingEdges,
+                                                EnglishGrammaticalRelations.COPULA);
+
+                SemanticGraphEdge dep = null;
+                if(cop == null && !root.tag().startsWith("V")) {
+                    boolean cont = false;
+                    for(SemanticGraphEdge edge: outgoingEdges) { //This makes the car necessary --> OC relation in xcomp, we can
generate a new clause out of this copular relation (the car, 'is', necessary). So far we are exiting
+                        if(DpUtils.isDep(edge) && edge.getDependent().tag().startsWith("V")) {
+                            dep = edge;
+                            cont = true;
+                            break;
+                        }
+                    }
+                    if(!cont)
+                        return;
+                }
+
+                // initialize clause
+                Clause clause = new Clause();
+                clause.setTree(depTree);
+                clause.setVerb(-1);
+                clause.setRoot(root);
+                if(ccs.size() > 1){
+//              SemanticGraph nSemanticGraph = new SemanticGraph(semanticGraph);
+                    SemanticGraph nSemanticGraph = SemanticGraphFactory.duplicateKeepNodes(semanticGraph);
+                    nSemanticGraph.setRoot(root);
+                    ProcessConjunctions.removeUnnecessary(nSemanticGraph, root);
+                    clause.setSemanticGraph(nSemanticGraph);
+                } else {
+                    clause.setSemanticGraph(semanticGraph);
+                }
+
+
+
+                Set<IndexedWord> exclude = null;
+                Set<IndexedWord> include = null;
+                if (cop != null) {
+                    clause.setCop(true);
+                    exclude = DpUtils.exclude(semanticGraph, EXCLUDE_RELATIONS_COMPLEMENT, root);
+                    include = DpUtils.exclude(semanticGraph, INCLUDE_RELATIONS_VERB, root);
+                } else {
+                    exclude = new HashSet<IndexedWord>();
+                }
+
+                // relative clause?
+                SemanticGraphEdge rcmod = DpUtils.findFirstOfRelation(incomingEdges,
+                                                EnglishGrammaticalRelations.RELATIVE_CLAUSE_MODIFIER);
+                SemanticGraphEdge poss = null;
+                if (rcmod != null)
+                    poss = DpUtils.findDescendantRelativeRelation(semanticGraph, root,
+                                                EnglishGrammaticalRelations.POSSESSION_MODIFIER);
+
+                // determine constituents of clause
+                //ArrayList<IndexedWord> coordinatedConjunctions = new ArrayList<IndexedWord>(); // to
```

```
+                // store
+                // potential
+                // conjunctions
+
+//------------------------Set verb or complement, and subject.-----------------------------------------------------
+
+
+
+                Constituent constRoot = null;
+                if (cop != null || (root.tag().startsWith("N") || root.tag().startsWith("J")) && dep != null) {   //He gets wet. The parser does
not recognize get, the second
+                    if(dep != null) {
+                        clause.getSemanticGraph().addEdge(root, dep.getDependent(), EnglishGrammaticalRelations.COPULA, 0, false);
+                        clause.getSemanticGraph().removeEdge(dep);
+                        outgoingEdges = clause.getSemanticGraph().getOutEdgesSorted(root);
+                        cop = DpUtils.findFirstOfRelationOrDescendent(outgoingEdges, EnglishGrammaticalRelations.COPULA);
+                        clause.setCop(true);
+                        exclude = DpUtils.exclude(clause.getSemanticGraph(), EXCLUDE_RELATIONS_COMPLEMENT, root);
+                        include = DpUtils.exclude(clause.getSemanticGraph(), INCLUDE_RELATIONS_VERB, root);
+                    }
+                    clause.setComplement(clause.getConstituents().size());
+                    constRoot = new StructuredConstituent(clause.getSemanticGraph(), root,
+                                    Collections.<IndexedWord> emptySet(), exclude, Constituent.Type.COMPLEMENT).build();
+                    clause.getConstituents().add(constRoot);
+
+                    clause.setVerb(clause.getConstituents().size());
+                    if (!partmod) {
+                        clause.getConstituents().add(new StructuredConstituent(clause.getSemanticGraph(), cop
+                                                .getDependent(), include, Collections.<IndexedWord> emptySet(),
+                                                Constituent.Type.VERB).build());
+                    } else {
+                        clause.getConstituents().add(new TextConstituent("be " + clauseRoot.word(),
+                                            Constituent.Type.VERB));
+                    }
+
+                } else {
+                    clause.setVerb(clause.getConstituents().size());
+                    if (!partmod) {
+                        constRoot = new StructuredConstituent(semanticGraph, root,
+                                        Collections.<IndexedWord> emptySet(), exclude, Constituent.Type.VERB).build();
+                    } else {
+                        constRoot = new TextConstituent("be " + clauseRoot.word(),
+                                        Constituent.Type.VERB);
+                    }
+
+                    clause.getConstituents().add(constRoot);
+                }
+
+                clause.setSubject(clause.getConstituents().size());
+                if (subject.tag().charAt(0) == 'W' && rcmod != null) {
+                    clause.getConstituents().add(createRelConstituent(semanticGraph, rcmod.getGovernor(),
+                                    Constituent.Type.SUBJECT));
+                    ((IndexedConstituent) constRoot).getExcludedVertexes().add(rcmod.getGovernor()      );
+                    rcmod = null;
+                } else if (poss != null && poss.getGovernor().equals(subject) && rcmod != null) {
+                    clause.getConstituents().add(createPossConstituent(semanticGraph, poss, rcmod, subject,
+                                    Constituent.Type.SUBJECT));
+                    rcmod = null;
+                } else if (partmod && subject.tag().charAt(0) == 'V') {
+                    List<SemanticGraphEdge> outsub = semanticGraph.getOutEdgesSorted(subject);
+                    SemanticGraphEdge sub = DpUtils.findFirstOfRelationOrDescendent(outsub,
+                                    EnglishGrammaticalRelations.SUBJECT);
+                    if (sub != null)
+                        clause.getConstituents().add(new StructuredConstituent(semanticGraph, sub
+                                                .getDependent(), Constituent.Type.SUBJECT).build());
+                    else
+                        clause.getConstituents().add(new StructuredConstituent(semanticGraph, subject,
+                                                Constituent.Type.SUBJECT).build());
+
```

```
+                } else
+                    clause.getConstituents().add(new StructuredConstituent(semanticGraph, subject,
+                                                    Constituent.Type.SUBJECT).build());
+
+                //If the clause comes from a partmod construction exclude necessary vertex
+                if (partmod) {
+                    ((IndexedConstituent) clause.getConstituents().get(clause.getSubject())).excludedVertexes
+                        .add(clauseRoot);
+                    // He is the man crying the whole day.
+                    List<SemanticGraphEdge> outsub = semanticGraph.getOutEdgesSorted(subject);
+                    SemanticGraphEdge coppm = DpUtils.findFirstOfRelationOrDescendent(outsub,
+                                                    EnglishGrammaticalRelations.COPULA);
+                    if (coppm != null) {
+                        ((IndexedConstituent) clause.getConstituents().get(clause.getSubject())).excludedVertexes
+                            .add(coppm.getDependent());
+                        SemanticGraphEdge spm = DpUtils.findFirstOfRelationOrDescendent(outsub,
+                                                        EnglishGrammaticalRelations.SUBJECT);
+                        ((IndexedConstituent) clause.getConstituents().get(clause.getSubject())).excludedVertexes
+                            .add(spm.getDependent());
+                    }
+
+                }
+
+                //------------------------Select constituents of the predicate---------------------------------------------------
+                for (SemanticGraphEdge outgoingEdge : outgoingEdges) {
+                    IndexedWord dependent = outgoingEdge.getDependent();
+
+                    // to avoid compl or mark in a main clause. "I doubt if she was sure whether this was important".
+                    if(DpUtils.isPassive(outgoingEdge)) {
+                        clause.setPassive(true);
+                    } else if (//DpUtils.isComplm(outgoingEdge) ||
+                        DpUtils.isMark(outgoingEdge)) {
+                        ((IndexedConstituent) constRoot).getExcludedVertexes().add(dependent);
+                        //Indirect Object
+                    } else if (DpUtils.isIobj(outgoingEdge)) {
+                        clause.getIobjects().add(clause.getConstituents().size());
+                        //If it is a relative clause headed by a relative pronoun.
+                        if (dependent.tag().charAt(0) == 'W' && rcmod != null) {
+                            clause.getConstituents().add(createRelConstituent(semanticGraph,
+                                                            rcmod.getGovernor(), Constituent.Type.IOBJ));
+                            ((IndexedConstituent) constRoot).getExcludedVertexes().add(dependent);
+                            rcmod = null;
+                            //to deal with the possessive relative pronoun
+                        } else if (poss != null && poss.getGovernor().equals(dependent)
+                              && rcmod != null) {
+                            clause.getConstituents().add(createPossConstituent(semanticGraph, poss, rcmod,
+                                                            dependent, Constituent.Type.IOBJ));
+                            rcmod = null;
+                            // "regular case"
+                        } else
+                            clause.getConstituents().add(new StructuredConstituent(semanticGraph, dependent,
+                                                            Constituent.Type.IOBJ).build());
+                        //Direct Object
+                    } else if (DpUtils.isDobj(outgoingEdge)) {
+                        clause.getDobjects().add(clause.getConstituents().size());
+                        if (dependent.tag().charAt(0) == 'W' && rcmod != null) {
+                            clause.getConstituents().add(createRelConstituent(semanticGraph,
+                                                            rcmod.getGovernor(), Constituent.Type.DOBJ));
+                            ((IndexedConstituent) constRoot).getExcludedVertexes().add(dependent);
+                            rcmod = null;
+                        } else if (poss != null && poss.getGovernor().equals(dependent)
+                              && rcmod != null) {
+                            clause.getConstituents().add(createPossConstituent(semanticGraph, poss, rcmod,
+                                                            dependent, Constituent.Type.DOBJ));
+                            rcmod = null;
+                        } else
+                            clause.getConstituents().add(new StructuredConstituent(semanticGraph, dependent,
+                                                            Constituent.Type.DOBJ).build());
+                        //CCOMPS
```

```
+                    } else if (DpUtils.isCcomp(outgoingEdge)) {
+                        clause.getCcomps().add(clause.getConstituents().size());
+                        clause.getConstituents().add(new StructuredConstituent(semanticGraph, dependent,
+                                                    Constituent.Type.CCOMP).build());
+                        //XCOMPS (Note: Need special treatment, they won't form a new clause so optional/obligatory constituents
+                        // are managed within the context of its parent clause)
+                    } else if (DpUtils.isXcomp(outgoingEdge)) {
+                        List<IndexedWord> xcomproots = new ArrayList<IndexedWord>();
+                        List<Clause> xcompclauses = new ArrayList<Clause>();
+                        IndexedWord xcompsubject = null;
+                        SemanticGraphEdge xcsub = DpUtils.findFirstOfRelationOrDescendent(
+                            semanticGraph.getOutEdgesSorted(outgoingEdge.getDependent()),
+                            EnglishGrammaticalRelations.SUBJECT);
+                        if (xcsub != null)
+                            xcompsubject = xcsub.getDependent();
+                        else
+                            xcompsubject = subject;
+                        //Need to identify the internal structure of the clause
+                        addNsubjClause(semanticGraph, depTree, options, xcomproots, xcompclauses, xcompsubject,
+                                outgoingEdge.getDependent(), false, true);
+                        for (Clause cl : xcompclauses) {
+                            if (xcsub == null) {
+                                int verb = cl.getVerb();
+                                ((IndexedConstituent) cl.getConstituents().get(verb)).additionalVertexes
+                                    .add(xcompsubject);
+                            }
+
+                            //do not include when there is no  verbal relation. This makes the car necessary. now OC relation appears as
xcomp
+                            boolean incXcompCl = true;
+                            if(!outgoingEdge.getDependent().tag().startsWith("V")){
+                                incXcompCl = false;
+                                for(SemanticGraphEdge edxc: semanticGraph.getOutEdgesSorted(outgoingEdge.getDependent())) {
+                                    if(edxc.getDependent().tag().startsWith("V")) {
+                                        incXcompCl = true;
+                                        break;
+                                    }
+                                }
+                            }
+
+
+                            if(includeXcomps && incXcompCl) {
+                                clauses.add(cl);
+                                roots.add(cl.getRoot());
+                                if (xcsub == null) {
+                                    SemanticGraphEdge sub =
+                                        DpUtils.findFirstOfRelationOrDescendent(semanticGraph.getIncomingEdgesSorted(subject),
EnglishGrammaticalRelations.SUBJECT);
+
+                                    GrammaticalRelation rel;
+                                    if(sub == null)
+                                        rel = EnglishGrammaticalRelations.SUBJECT;
+                                    else
+                                        rel = sub.getRelation();
+
+                                    cl.getSemanticGraph().addEdge(cl.getRoot(), xcompsubject, rel,
+                                                0, true);
+                                }
+                            }
+                            excludeVertexes(cl);
+                        }
+                        clause.getXcomps().add(clause.getConstituents().size());
+                        clause.getConstituents().add(new XcompConstituent(semanticGraph, dependent,
+                                                Constituent.Type.XCOMP, xcompclauses));
+
+
+
+                        //Adjective complement
+                    } else if (DpUtils.isAcomp(outgoingEdge)) {
```

```
+                           clause.getAcomps().add(clause.getConstituents().size());
+                           clause.getConstituents().add(new StructuredConstituent(semanticGraph, dependent,
+                                                   Constituent.Type.ACOMP).build());
+                           //Various Adverbials
+                       } else if ((DpUtils.isAnyPrep(outgoingEdge)
+                               || DpUtils.isPobj(outgoingEdge)
+                               || DpUtils.isTmod(outgoingEdge)
+                               || DpUtils.isAdvcl(outgoingEdge)
+                               || DpUtils.isNpadvmod(outgoingEdge)
+                               //|| DpUtils.isPurpcl(outgoingEdge)
+                                    )
+
+                           ) {
+                           if(rcmod != null && DpUtils.findRelClause(semanticGraph.getOutEdgesSorted(dependent))) {
+                                processRel(outgoingEdge, semanticGraph, dependent, rcmod, clause);
+                                rcmod = null;
+                           } else {
+                                int constint = clause.getConstituents().size();
+                                clause.getAdverbials().add(constint);
+                                clause.getConstituents().add(new StructuredConstituent(
+                                                        semanticGraph, dependent,
+                                                        Constituent.Type.ADVERBIAL).build());
+                           }
+                           //Advmod
+                       } else if (DpUtils.isAdvmod(outgoingEdge)) {
+                           int constint = clause.getConstituents().size();
+                           clause.getAdverbials().add(constint);
+                           clause.getConstituents().add(new StructuredConstituent(semanticGraph, dependent,
+                                                   Constituent.Type.ADVERBIAL).build());
+                           //Partmod
+                           //} else if (DpUtils.isPartMod(outgoingEdge)) {
+                           //    int constint = clause.getConstituents().size();
+                           //    clause.getAdverbials().add(constint);
+                           //    clause.getConstituents().add(new StructuredConstituent(semanticGraph, dependent,
+                           //          Constituent.Type.ADVERBIAL).build());
+                           //Rel appears in certain cases when relative pronouns act as prepositional objects "I saw the house in which I
grew".
+                           // We generate a new clause out of the relative clause
+                           //} else if (DpUtils.isRel(outgoingEdge)) {
+                           processRel(outgoingEdge, semanticGraph, dependent, rcmod, clause);
+                           rcmod = null;
+
+                           //To process passive voice (!Not done here)
+                           // } else if (DpUtils.isAgent(outgoingEdge))
+                           //     clause.agent = dependent;
+                           // else if (DpUtils.isMark(outgoingEdge) || DpUtils.isComplm(outgoingEdge)) {
+                           // clause.subordinateConjunction = dependent;
+                       } else if (DpUtils.isExpl(outgoingEdge))
+                           clause.setType(Clause.Type.EXISTENTIAL);
+                       //  else if (options.processCcAllVerbs && DpUtils.isAnyConj(outgoingEdge))
+                       //    coordinatedConjunctions.add(dependent);
+                   }
+
+                   //------------------------To process relative clauses with implicit (zero) relative pronoun-------------------------
+                   if (rcmod != null) { //"I saw the house I grew up in", "I saw
+                       // the house I like", "I saw the man I gave the book" ...
+                       Constituent candidate = searchCandidateAdverbial(clause);
+                       if (candidate != null) {
+                           // SemanticGraph newSemanticGraph = new SemanticGraph(
+                           //      ((IndexedConstituent) candidate).getSemanticGraph());
+                           SemanticGraph newSemanticGraph = SemanticGraphFactory.duplicateKeepNodes(((IndexedConstituent)
candidate).getSemanticGraph());
+                           IndexedConstituent tmpconst = createRelConstituent(newSemanticGraph,
+                                                   rcmod.getGovernor(), Constituent.Type.ADVERBIAL);
+                           newSemanticGraph.addEdge(((IndexedConstituent) candidate).getRoot(),
+                                   rcmod.getGovernor(), EnglishGrammaticalRelations.PREPOSITIONAL_OBJECT,
+                                   rcmod.getWeight(), false);
+                           ((IndexedConstituent) candidate).getExcludedVertexes().addAll(
+                               tmpconst.getExcludedVertexes());
```

```
+                        ((IndexedConstituent) candidate).setSemanticGraph(newSemanticGraph);
+                        rcmod = null;
+                    } else if (DpUtils.findFirstOfRelation(outgoingEdges,
+                                    EnglishGrammaticalRelations.DIRECT_OBJECT) == null) {
+                        clause.getDobjects().add(clause.getConstituents().size());
+                        clause.getConstituents().add(createRelConstituent(semanticGraph,
+                                        rcmod.getGovernor(), Constituent.Type.DOBJ));
+                        rcmod = null;
+                    } else if (DpUtils.findFirstOfRelation(outgoingEdges,
+                                    EnglishGrammaticalRelations.INDIRECT_OBJECT) == null) {
+                        clause.getIobjects().add(clause.getConstituents().size());
+                        clause.getConstituents().add(createRelConstituent(semanticGraph,
+                                        rcmod.getGovernor(), Constituent.Type.IOBJ));
+                        rcmod = null;
+                    }
+                }
+
+//------------------------------------------------------------------------------------------------------------------
+                //To deal with parataxis
+                SemanticGraphEdge parataxis = DpUtils.findFirstOfRelation(incomingEdges,
+                                EnglishGrammaticalRelations.PARATAXIS);
+                if (parataxis != null && clause.getConstituents().size() < 3) {
+                    addParataxisClause(semanticGraph, clauses, depTree, parataxis.getGovernor(), parataxis.getDependent(),
+                            roots);
+                    return; // to avoid generating (John, said) in "My dog, John said, is great" //To
+                    // deal with the type of parataxis. Parataxis are either like in the example
+                    // above or subclauses comming from ":" or ";" this is here because is
+                    // difficult to identify the type upfront. Otherwise we can count the potential
+                    // constituents upfront and move this up.
+                }
+
+
+                //Detect type and mantain clause lists
+                roots.add(root);
+                if (!partmod) {
+                    //            clause.detectType(options);
+                } else {
+                    //            clause.setType(Clause.Type.SVA);
+                }
+                clauses.add(clause);
+            }
+        }
+
+        /** Process relation rel, it creates a new clause out of the relative clause
+         * @param outgoingEdge The rel labeled edge
+         * @param semanticGraph The semantic graph
+         * @param dependent The dependent of the relation
+         * @param rcmod The relative clause modifier of the relation refered by rel
+         * @param clause A clause*/
+        public static void processRel(SemanticGraphEdge outgoingEdge, SemanticGraph semanticGraph, IndexedWord dependent,
SemanticGraphEdge rcmod, Clause clause) {
+            //SemanticGraph newSemanticGraph = new SemanticGraph(semanticGraph);
+            SemanticGraph newSemanticGraph = SemanticGraphFactory.duplicateKeepNodes(semanticGraph);
+            List<SemanticGraphEdge> outdep = newSemanticGraph.getOutEdgesSorted(dependent);
+            SemanticGraphEdge pobed = DpUtils.findFirstOfRelation(outdep,
+                                    EnglishGrammaticalRelations.PREPOSITIONAL_OBJECT);
+
+            SemanticGraphEdge posspobj = null;
+            if (pobed != null && pobed.getDependent().tag().charAt(0) != 'W') {
+                List<SemanticGraphEdge> outpobj = newSemanticGraph
+                        .getOutEdgesSorted(dependent);
+                posspobj = DpUtils.findFirstOfRelation(outpobj,
+                                    EnglishGrammaticalRelations.POSSESSION_MODIFIER);
+            }
+
+            if (pobed != null && pobed.getDependent().tag().charAt(0) == 'W'
+               && rcmod != null) {
+                newSemanticGraph.addEdge(dependent, rcmod.getGovernor(),
+                                EnglishGrammaticalRelations.PREPOSITIONAL_OBJECT,
```

```
+                                   pobed.getWeight(), false);
+                newSemanticGraph.removeEdge(pobed);
+                int constint = clause.getConstituents().size();
+                clause.getAdverbials().add(constint);
+                clause.getConstituents().add(createRelConstituent(newSemanticGraph,
+                                                rcmod.getGovernor(), Constituent.Type.SUBJECT));
+                ((IndexedConstituent) clause.getConstituents().get(constint))
+                     .setRoot(dependent);
+                clause.setRelativeAdverbial(true);
+                rcmod = null;
+           } else if (pobed != null && posspobj != null && rcmod != null) {
+                newSemanticGraph.addEdge(posspobj.getGovernor(), rcmod.getGovernor(),
+                                   EnglishGrammaticalRelations.POSSESSION_MODIFIER,
+                                   posspobj.getWeight(), false);
+                newSemanticGraph.removeEdge(posspobj);
+                int constint = clause.getConstituents().size();
+                clause.getAdverbials().add(constint);
+                // search pobj copy edge.
+                clause.getConstituents().add(createRelConstituent(newSemanticGraph,
+                                                rcmod.getGovernor(), Constituent.Type.SUBJECT));
+                ((IndexedConstituent) clause.getConstituents().get(constint))
+                     .setRoot(dependent);
+                clause.setRelativeAdverbial(true);
+           }
+
+      }
+
+      /** Finds the adverbial to which the relative clause is referring to*/
+      private static Constituent searchCandidateAdverbial(Clause clause) {
+           for (Constituent c : clause.getConstituents()) {
+                IndexedWord root = ((IndexedConstituent) c).getRoot();
+                if (root.tag().equals("IN")
+                   && !((IndexedConstituent) c).getSemanticGraph().hasChildren(root))
+                     return c;
+           }
+           return null;
+      }
+
+      /** Creates a constituent for a possessive relative clause
+       * @param semanticGraph The semantic graph
+       * @param poss The edge referring to the possessive relation
+       * @param rcmod The relative clause modifier of the relation
+       * @param constGovernor The root of the constituent
+       * @param type The type of the constituent*/
+      private static Constituent createPossConstituent(SemanticGraph semanticGraph,
+                                     SemanticGraphEdge poss, SemanticGraphEdge rcmod, IndexedWord constGovernor, Constituent.Type
type) {
+
+           //SemanticGraph newSemanticGraph = new SemanticGraph(semanticGraph);
+           SemanticGraph newSemanticGraph = SemanticGraphFactory.duplicateKeepNodes(semanticGraph);
+           double weight = poss.getWeight();
+           newSemanticGraph.addEdge(poss.getGovernor(), rcmod.getGovernor(),
+                          EnglishGrammaticalRelations.POSSESSION_MODIFIER, weight, false);
+           Set<IndexedWord> exclude = DpUtils.exclude(newSemanticGraph, EXCLUDE_RELATIONS_COMPLEMENT,
+                                     rcmod.getGovernor());
+           newSemanticGraph.removeEdge(poss);
+           newSemanticGraph.removeEdge(rcmod);
+           return new StructuredConstituent(newSemanticGraph, constGovernor,
+                                   Collections.<IndexedWord> emptySet(), exclude, type).build();
+      }
+
+      /** Creates a constituent for the relative clause implied by rel
+       * @param semanticGraph The semantic graph
+       * @param root The root of the constituent
+       * @param type The type of the constituent*/
+      private static IndexedConstituent createRelConstituent(SemanticGraph semanticGraph,
+                                       IndexedWord root, Constituent.Type type) {
+
+           List<SemanticGraphEdge> outrcmod = semanticGraph.getOutEdgesSorted(root);
```

```
+            SemanticGraphEdge rccop = DpUtils.findFirstOfRelation(outrcmod,
+                                                  EnglishGrammaticalRelations.COPULA);
+            if (rccop != null) {
+                 Set<IndexedWord> excludercmod = DpUtils.exclude(semanticGraph,
+                                                   EXCLUDE_RELATIONS_COMPLEMENT, root);
+                 return new StructuredConstituent(semanticGraph, root,
+                                         Collections.<IndexedWord> emptySet(), excludercmod, type).build();
+            } else
+                 return new StructuredConstituent(semanticGraph, root, type).build();
+       }
+
+       /** Generates a clause from an apposition
+        * @param subject The subject of the clause (first argument of the appos relation)
+        * @param object  The object of the clause (second argument of the appos relation)*/
+       private static void addApposClause(Tree depTree, SemanticGraph semanticGraph, Options options, List<Clause> clauses,
IndexedWord subject, IndexedWord object) {
+            Clause clause = new Clause();
+            clause.setSubject(0);
+            clause.setVerb(1);
+            clause.setComplement(2);
+            clause.setTree(depTree);
+            clause.getConstituents().add(new StructuredConstituent(semanticGraph, subject,
+                                                  Constituent.Type.SUBJECT).build());
+            clause.getConstituents().add(new TextConstituent(options.appositionVerb,
+                                               Constituent.Type.VERB));
+            clause.getConstituents().add(new StructuredConstituent(semanticGraph, object,
+                                                  Constituent.Type.COMPLEMENT).build());
+            clause.setType(Clause.Type.SVC);
+            clauses.add(clause);
+       }
+
+       /** Generates a clause from a possessive relation
+        * @param subject The subject of the clause
+        * @param object  The object of the clause */
+       private static void addPossessiveClause(Tree depTree, SemanticGraph semanticGraph, Options options, List<Clause> clauses,
IndexedWord subject,
+                                          IndexedWord object) {
+            Clause clause = new Clause();
+            //SemanticGraph newSemanticGraph = new SemanticGraph(semanticGraph);
+            SemanticGraph newSemanticGraph = SemanticGraphFactory.duplicateKeepNodes(semanticGraph);
+            clause.setSubject(0);
+            clause.setVerb(1);
+            clause.getDobjects().add(2);
+            clause.setTree(depTree);
+            Set<IndexedWord> excludesub = new TreeSet<IndexedWord>();
+            Set<IndexedWord> excludeobj = new TreeSet<IndexedWord>();
+
+            excludeobj.add(subject);
+            List<SemanticGraphEdge> outedobj = newSemanticGraph.getOutEdgesSorted(object);
+            excludeVertexPoss(outedobj, excludeobj, options);
+
+            SemanticGraphEdge rcmod = null;
+            if (subject.tag().charAt(0) == 'W') {
+                 IndexedWord root = newSemanticGraph.getParent(object);
+                 if (root.tag().equals("IN"))
+                      root = newSemanticGraph.getParent(root); // "I saw the man in whose wife I trust"
+                 List<SemanticGraphEdge> inedges = newSemanticGraph.getIncomingEdgesSorted(root);
+                 rcmod = DpUtils.findFirstOfRelation(inedges,
+                                          EnglishGrammaticalRelations.RELATIVE_CLAUSE_MODIFIER);
+            } else {
+                 List<SemanticGraphEdge> outedges = newSemanticGraph.getOutEdgesSorted(subject);
+                 SemanticGraphEdge ps = DpUtils.findFirstOfRelation(outedges,
+                                                   EnglishGrammaticalRelations.POSSESSIVE_MODIFIER);
+                 if (ps != null)
+                      excludesub.add(ps.getDependent());
+            }
+
+            if (rcmod != null) {
+                 clause.getConstituents().add(createRelConstituent(newSemanticGraph, rcmod.getGovernor(),
```

```
+                                                   Constituent.Type.SUBJECT));
+                ((IndexedConstituent) clause.getConstituents().get(0)).getExcludedVertexes().addAll(
+                    excludesub); // to avoid the s in  "Bill's clothes are great".
+            } else {
+                clause.getConstituents().add(new StructuredConstituent(newSemanticGraph, subject, Collections
+                                                    .<IndexedWord> emptySet(), excludesub, Constituent.Type.SUBJECT).build());
+            }
+            clause.getConstituents().add(new TextConstituent(options.possessiveVerb,
+                                            Constituent.Type.VERB));
+            clause.getConstituents().add(new StructuredConstituent(newSemanticGraph, object, Collections
+                                            .<IndexedWord> emptySet(), excludeobj, Constituent.Type.DOBJ).build());
+            clause.setType(Clause.Type.SVO);
+            clauses.add(clause);
+        }
+
+        /** Excludes vertexes for the object of a "possessive clause"
+         * @param outedobj relations to be examined for exclusion
+         * @param excludeobj The vertexes to be excluded*/
+        private static void excludeVertexPoss(List<SemanticGraphEdge> outedobj,
+                                    Set<IndexedWord> excludeobj, Options options) {
+            for (SemanticGraphEdge ed : outedobj) {
+                if (DpUtils.isAdvcl(ed)
+                    || DpUtils.isAdvmod(ed)
+                    || DpUtils.isAnyObj(ed) // currently everything is
+                    // excluded except prep and infmod
+                    || DpUtils.isAnySubj(ed) || DpUtils.isAux(ed) || DpUtils.isCop(ed)
+                    || DpUtils.isTmod(ed) || DpUtils.isAnyConj(ed)
+                    && options.processCcNonVerbs)
+                    excludeobj.add(ed.getDependent());
+            }
+
+        }
+
+        /** Creates a clause from a partmod relation
+         * @param subject The subject of the clause
+         * @param object  The object of the clause
+         * @param roots List of clause roots*/
+        private static void addPartmodClause(SemanticGraph semanticGraph, Tree depTree, Options options, List<Clause>
clauses,IndexedWord subject, IndexedWord verb,
+                                    List<IndexedWord> roots, boolean includeXcomps) {
+            IndexedWord partmodsub = subject;
+            addNsubjClause(semanticGraph, depTree, options, roots, clauses, partmodsub, verb, true, includeXcomps);
+        }
+
+        /** Creates a clause from a parataxis relation
+     * @param root Head of the parataxis relation
+     * @param parroot  Dependent of the parataxis relation
+     * @param roots List of clause roots*/
+    private static void addParataxisClause(SemanticGraph semanticGraph, List<Clause> clauses, Tree depTree, IndexedWord root,
IndexedWord parroot,
+            List<IndexedWord> roots) {
+        Constituent verb = new StructuredConstituent(semanticGraph, parroot, Constituent.Type.VERB).build();
+        List<SemanticGraphEdge> outedges = semanticGraph.getOutEdgesSorted(parroot);
+        SemanticGraphEdge subject = DpUtils.findFirstOfRelationOrDescendent(outedges,
+            EnglishGrammaticalRelations.SUBJECT);
+        if (subject != null) {
+            Constituent subjectConst = new StructuredConstituent(semanticGraph,
+                subject.getDependent(), Constituent.Type.SUBJECT).build();
+            Constituent object = new StructuredConstituent(semanticGraph, root, Constituent.Type.DOBJ).build();
+            ((IndexedConstituent) object).excludedVertexes.add(parroot);
+            Clause clause = new Clause();
+            clause.setTree(depTree);
+            clause.setSemanticGraph(semanticGraph);
+            clause.setSubject(0);
+            clause.setVerb(1);
+            clause.setRoot(parroot);
+            clause.getDobjects().add(2);
+            clause.getConstituents().add(subjectConst);
+            clause.getConstituents().add(verb);
```

```
+          clause.getConstituents().add(object);
+          clause.setType(Clause.Type.SVO);
+          clauses.add(clause);
+          roots.add(parroot);
+
+      }
+
+   }
+}
diff --git a/src/clausie/Constituent.java b/src/clausie/Constituent.java
new file mode 100644
index 0000000..1ae961a
--- /dev/null
+++ b/src/clausie/Constituent.java
@@ -0,0 +1,80 @@
+package clausie;
+
+/** A constituent of a clause. */
+public abstract class Constituent {
+
+   // -- types ------------------------------------------------------------------------------
+
+   /** Constituent types */
+   public enum Type {
+          SUBJECT("S"),
+          VERB("V"),
+          DOBJ("O"),
+          IOBJ("IO"),
+          COMPLEMENT("C"),
+          CCOMP("CCOMP"),
+          XCOMP("XCOMP"),
+          ACOMP("ACOMP"),
+          ADVERBIAL("A"),
+          UNKOWN("UNKNOWN");
+
+          private final String name;
+
+          private Type(String s) {
+                name = s;
+          }
+
+          public String toString() {
+                return name;
+          }
+   };
+
+   /** Constituent flags */
+   public enum Flag {
+          REQUIRED("!"),
+          OPTIONAL("?"),
+          IGNORE("-");
+
+          private final String name;
+
+          private Flag(String s) {
+                name = s;
+          }
+
+          public String toString() {
+                return name;
+          }
+   };
+
+
+      // -- member variables ------------------------------------------------------------------
+
+      /** Type of this constituent */
+      public Type type;
+
```

```
+
+	// -- construction ---------------------------------------------------------------
+
+	/** Constructs a constituent of the specified type. */
+	public Constituent(Type type) {
+		this.type = type;
+	}
+
+	/** Constructs a constituent of unknown type. */
+	public Constituent() {
+		this.type = Type.UNKOWN;
+	}
+
+	// -- getters/setters ------------------------------------------------------------
+
+	/** Returns the type of this constituent. */
+	public Type getType() {
+		return type;
+	}
+
+
+	// -- utility methods ------------------------------------------------------------
+
+   /** Returns a textual representation of the root word of this constituent. */
+   public abstract String rootString();
+}
diff --git a/src/clausie/DefaultPropositionGenerator.java b/src/clausie/DefaultPropositionGenerator.java
new file mode 100644
index 0000000..6e54512
--- /dev/null
+++ b/src/clausie/DefaultPropositionGenerator.java
@@ -0,0 +1,97 @@
+package clausie;
+
+import java.util.ArrayList;
+import java.util.List;
+import java.util.SortedSet;
+import java.util.TreeSet;
+
+import clausie.Constituent.Flag;
+
+
+/** Currently the default proposition generator generates 3-ary propositions out of a clause.
+ *
+ * @date $LastChangedDate: 2013-08-26 15:31:37 +0200 (Mon, 26 Aug 2013) $
+ * @version $LastChangedRevision: 977 $ */
+public class DefaultPropositionGenerator extends PropositionGenerator {
+	public DefaultPropositionGenerator(Options options) {
+		super(options);
+	}
+
+	@Override
+	public void generate(List<Proposition> result, Clause clause,
+			List<Boolean> include) {
+		Proposition proposition = new Proposition();
+		List<Proposition> propositions = new ArrayList<Proposition>();
+
+		// process subject
+		if (clause.getSubject() > -1 && include.get(clause.getSubject())) { // subject is -1 when there is an xcomp
+			proposition.constituents.add( generate(clause, clause.getSubject()) );
+		} else {
+			//throw new IllegalArgumentException();
+		}
+
+		// process verb
+		if (include.get(clause.getVerb())) {
+			proposition.constituents.add( generate(clause, clause.getVerb()) );
+		} else {
+			throw new IllegalArgumentException();
```

```
+            }
+
+            propositions.add(proposition);
+
+            // process arguments
+            SortedSet<Integer> sortedIndexes = new TreeSet<Integer>();
+            sortedIndexes.addAll(clause.getIobjects());
+            sortedIndexes.addAll(clause.getDobjects());
+            sortedIndexes.addAll(clause.getXcomps());
+            sortedIndexes.addAll(clause.getCcomps());
+            sortedIndexes.addAll(clause.getAcomps());
+            sortedIndexes.addAll(clause.getAdverbials());
+            if (clause.getComplement() >= 0)
+                sortedIndexes.add(clause.getComplement());
+            for(Integer index: sortedIndexes) {
+                if (clause.getConstituents().get(clause.getVerb()) instanceof IndexedConstituent && clause.getAdverbials().contains(index)
&& ((IndexedConstituent)clause.getConstituents().get(index)).getRoot().index() <
((IndexedConstituent)clause.getConstituents().get(clause.getVerb())).getRoot().index()) continue;
+                for(Proposition p: propositions) {
+                    if (include.get(index)) {
+                        p.constituents.add( generate(clause, index) );
+                    }
+                }
+            }
+
+            // process adverbials  before verb
+            sortedIndexes.clear();
+            sortedIndexes.addAll(clause.getAdverbials());
+            for (Integer index : sortedIndexes) {
+                if (clause.getConstituents().get(clause.getVerb()) instanceof TextConstituent ||
((IndexedConstituent)clause.getConstituents().get(index)).getRoot().index() >
((IndexedConstituent)clause.getConstituents().get(clause.getVerb())).getRoot().index()) break;
+                if (include.get(index)) {
+                    for(Proposition p: propositions) {
+                        p.constituents.add( generate(clause, index) );
+                        if (clause.getFlag(index, options).equals(Flag.OPTIONAL)) {
+                            p.optional.add(p.constituents.size());
+                        }
+                    }
+                }
+            }
+
+            // make 3-ary if needed
+            if (!options.nary ) {
+                for(Proposition p: propositions) {
+                    p.optional.clear();
+                    if (p.constituents.size() > 3) {
+                        StringBuilder arg = new StringBuilder();
+                        for (int i=2; i<p.constituents.size(); i++) {
+                            if (i>2) arg.append(" ");
+                            arg.append(p.constituents.get(i));
+                        }
+                        p.constituents.set(2,  arg.toString());
+                        for (int i=p.constituents.size()-1; i>2; i--) {
+                            p.constituents.remove(i);
+                        }
+                    }
+                }
+            }
+
+                    // we are done
+                    result.addAll(propositions);
+         }
+}
diff --git a/src/clausie/Dictionary.java b/src/clausie/Dictionary.java
new file mode 100644
index 0000000..b39f6e0
--- /dev/null
+++ b/src/clausie/Dictionary.java
```

```
@@ -0,0 +1,56 @@
+package clausie;
+
+import java.io.DataInput;
+import java.io.DataInputStream;
+import java.io.IOException;
+import java.io.InputStream;
+import java.util.HashSet;
+import java.util.Set;
+
+import edu.stanford.nlp.ling.IndexedWord;
+
+/**A dictionary stores a set of strings.
+ *
+ * @date $LastChangedDate: 2013-04-23 12:03:16 +0200 (Tue, 23 Apr 2013) $
+ * @version $LastChangedRevision: 735 $ */
+public class Dictionary {
+
+	/** Stores the strings */
+	public Set<String> words = new HashSet<String>();
+
+	public Dictionary() {
+	}
+
+	public int size() {
+		return words.size();
+	}
+
+	public boolean contains(String word) {
+		return words.contains(word);
+	}
+
+	public boolean contains(IndexedWord word) {
+		return words.contains( word.lemma() );
+	}
+
+	/** Loads the dictionary out of an {@link InputStream}. Each line
+	 * of the original file should contain an entry to the dictionary */
+	public void load(InputStream in) throws IOException {
+		DataInput data = new DataInputStream(in);
+		String line = data.readLine();
+		while (line != null) {
+			line = line.trim();
+			if (line.length() > 0) { // treat everything else as comments
+				if (Character.isLetter(line.charAt(0))) {
+					words.add(line);
+				}
+			}
+			line = data.readLine();
+		}
+	}
+
+	public Set<String> words() {
+		return words;
+	}
+
+}
diff --git a/src/clausie/DpUtils.java b/src/clausie/DpUtils.java
new file mode 100644
index 0000000..abd5a7b
--- /dev/null
+++ b/src/clausie/DpUtils.java
@@ -0,0 +1,561 @@
+package clausie;
+
+import java.util.ArrayList;
+import java.util.Collection;
+import java.util.Collections;
+import java.util.HashSet;
```

```
+import java.util.List;
+import java.util.Set;
+import java.util.TreeSet;
+
+import edu.stanford.nlp.ling.IndexedWord;
+import edu.stanford.nlp.semgraph.SemanticGraph;
+import edu.stanford.nlp.semgraph.SemanticGraphEdge;
+import edu.stanford.nlp.trees.EnglishGrammaticalRelations;
+import edu.stanford.nlp.trees.GrammaticalRelation;
+import edu.stanford.nlp.trees.Tree;
+
+
+/** This class provides a set of utilities to work with {@link SemanticGraph}
+ * For details on the Dependency parser @see <a href="nlp.stanford.edu/software/dependencies_manual.pdf">the Stanford Parser
manual
+ *
+ * @date $LastChangedDate: 2013-12-02 15:45:41 +0100 (Mon, 02 Dec 2013) $
+ * @version $LastChangedRevision: 1182 $ */
+public class DpUtils {
+
+	/** Finds the first occurrence of a grammatical relation in a set of edges */
+	public static SemanticGraphEdge findFirstOfRelation(List<SemanticGraphEdge> edges,
+	                                    GrammaticalRelation rel) {
+		for (SemanticGraphEdge e : edges) {
+			if (rel.equals(e.getRelation())) {
+				return e;
+			}
+		}
+		return null;
+	}
+
+	/** Finds the first occurrence of a grammatical relation or its descendants in a set of edges */
+	public static SemanticGraphEdge findFirstOfRelationOrDescendent(List<SemanticGraphEdge> edges,
+	                                        GrammaticalRelation rel) {
+		for (SemanticGraphEdge e : edges) {
+			if (rel.isAncestor(e.getRelation())) {
+				return e;
+			}
+		}
+		return null;
+	}
+
+	/** Finds the first occurrence of a grammatical relation or its descendants for a relative pronoun */
+	public static SemanticGraphEdge findDescendantRelativeRelation(SemanticGraph semanticGraph,
+	                                        IndexedWord root, GrammaticalRelation rel) {
+		List<SemanticGraphEdge> explored = new ArrayList<SemanticGraphEdge>();
+		return findDescendantRelativeRelation(semanticGraph, root, rel, explored);
+	}
+
+	public static SemanticGraphEdge findDescendantRelativeRelation(SemanticGraph semanticGraph,
+	                                        IndexedWord root, GrammaticalRelation rel, List<SemanticGraphEdge> explored) {
+		List<SemanticGraphEdge> outedges = semanticGraph.getOutEdgesSorted(root);
+		for (SemanticGraphEdge e : outedges) {
+			if(explored.contains(e))
+				continue;
+			if (e.getDependent().tag().charAt(0) == 'W' && rel.isAncestor(e.getRelation())) {
+				return e;
+			} else {
+				explored.add(e); //sometimes there are cycles probably a bug in the parser
+				return findDescendantRelativeRelation(semanticGraph, e.getDependent(), rel, explored);
+			}
+		}
+		return null;
+	}
+
+	/** Finds all occurrences of a grammatical relation or its descendants in a list of edges */
+	public static List<SemanticGraphEdge> getEdges(List<SemanticGraphEdge> edges,
+	                                GrammaticalRelation rel) {
+		List<SemanticGraphEdge> result = new ArrayList<SemanticGraphEdge>();
```

```
+            for (SemanticGraphEdge e : edges) {
+                if (rel.isAncestor(e.getRelation())) {
+                    result.add(e);
+                }
+            }
+            return result;
+        }
+
+        /** Finds all occurrences of a list of grammatical relation or its descendants in a list of edges */
+        public static List<SemanticGraphEdge> getEdges(List<SemanticGraphEdge> edges,
+                                        List<GrammaticalRelation> relations) {
+            List<SemanticGraphEdge> result = new ArrayList<SemanticGraphEdge>();
+            for (SemanticGraphEdge e : edges) {
+                for(GrammaticalRelation rel: relations) {
+                    if (rel.isAncestor(e.getRelation())) {
+                        result.add(e);
+                    }
+                }
+            }
+            return result;
+        }
+
+        public static Set<IndexedWord> getRelevantWords(IndexedWord root, SemanticGraph semanticGraph, Tree tree,
+                                        GrammaticalRelation rel, boolean down) {
+            Set<IndexedWord> ignore = new HashSet<IndexedWord>();
+            return getRelevantWords(root, semanticGraph, tree, rel, down, ignore);
+
+        }
+
+        public static Set<IndexedWord> getRelevantWords(IndexedWord root, SemanticGraph semanticGraph, Tree tree,
+                                        GrammaticalRelation rel, boolean down, Set<IndexedWord> ignore) {
+
+            Set<IndexedWord> result = new HashSet<IndexedWord>();
+            List<SemanticGraphEdge> edges;
+            if(down) { //Every parent from the main conj is also a parent for the conjs, not every child of the parent is a child of the conjs
+                edges = semanticGraph.getOutEdgesSorted(root);
+            } else {
+                edges = semanticGraph.getIncomingEdgesSorted(root);
+            }
+
+            for (SemanticGraphEdge e : edges) {
+                if (rel.isAncestor(e.getRelation())) {
+                    IndexedWord toAdd;
+                    if(down) {
+                        toAdd = e.getDependent();
+                    } else {
+                        toAdd = e.getGovernor();
+                    }
+                    if(ignore.contains(toAdd))
+                        continue;
+
+                    result.add(toAdd);
+                    ignore.add(toAdd);
+                    result.addAll(getRelevantWords(toAdd, semanticGraph, tree,
+                                            EnglishGrammaticalRelations.CONJUNCT, true, ignore));
+                }
+            }
+
+            ignore.addAll(result);
+            List<SemanticGraphEdge> parents = getEdges(semanticGraph.getIncomingEdgesSorted(root),
EnglishGrammaticalRelations.CONJUNCT);
+            for(SemanticGraphEdge edge: parents) {
+                if(ignore.contains(edge.getGovernor()))
+                    continue;
+                for(IndexedWord element: getRelevantWords(edge.getGovernor(), semanticGraph, tree, rel, down, ignore)) {
+                    if(down) {
+                        if(ProcessConjunctions.isDescendant(tree, semanticGraph, root, edge.getGovernor(), element)) {
+                            result.add(element);
+                        }
```

```
+                     } else {
+                         result.add(element);
+                     }
+                 }
+             }
+             return result;
+     }
+
+     /** Checks if a given grammatical relation is contained in a set of edges */
+     public static boolean containsRelation(List<SemanticGraphEdge> edges, GrammaticalRelation rel) {
+         return findFirstOfRelation(edges, rel) != null;
+     }
+
+     /** Checks if a given edge holds a subject relation*/
+     public static boolean isAnySubj(SemanticGraphEdge edge) {
+         return EnglishGrammaticalRelations.SUBJECT.isAncestor(edge.getRelation());
+     }
+
+     /** Checks if a given edge holds an adjective modifier relation*/
+     public static boolean isAMod(SemanticGraphEdge edge) {
+         return EnglishGrammaticalRelations.ADJECTIVAL_MODIFIER.equals(edge.getRelation());
+     }
+
+     /** Checks if a given edge holds a noun compound relation*/
+     public static boolean isNN(SemanticGraphEdge edge) {
+         return EnglishGrammaticalRelations.NOUN_COMPOUND_MODIFIER.equals(edge.getRelation());
+     }
+
+     /** Checks if a given edge holds a nominal subject relation*/
+     public static boolean isNsubj(SemanticGraphEdge edge) {
+         return EnglishGrammaticalRelations.NOMINAL_SUBJECT.equals(edge.getRelation());
+     }
+
+     /** Checks if a given edge holds a clausal subject relation*/
+     public static boolean isCsubj(SemanticGraphEdge edge) {
+         return EnglishGrammaticalRelations.CLAUSAL_SUBJECT.equals(edge.getRelation());
+     }
+
+     /** Checks if a given edge holds a clausal passive subject relation*/
+     public static boolean isCsubjpass(SemanticGraphEdge edge) {
+         return EnglishGrammaticalRelations.CLAUSAL_PASSIVE_SUBJECT.equals(edge.getRelation());
+     }
+
+     /** Checks if a given edge holds a nominal passive subject relation*/
+     public static boolean isNsubjpass(SemanticGraphEdge edge) {
+         return EnglishGrammaticalRelations.NOMINAL_PASSIVE_SUBJECT.equals(edge.getRelation());
+     }
+
+     /** Checks if a given edge holds an external subject relation of an xcomp relation */
+     public static boolean isXsubj(SemanticGraphEdge edge) {
+         return edge.toString().equals("xsubj");
+     }
+
+     /** Checks if a given edge holds an object relation */
+     public static boolean isAnyObj(SemanticGraphEdge edge) {
+         return EnglishGrammaticalRelations.OBJECT.isAncestor(edge.getRelation());
+     }
+
+     /** Checks if a given edge holds a prepositional object relation*/
+     public static boolean isPobj(SemanticGraphEdge edge) {
+         return EnglishGrammaticalRelations.PREPOSITIONAL_OBJECT.equals(edge.getRelation());
+     }
+
+     /** Checks if a given edge holds a direct object relation */
+     public static boolean isDobj(SemanticGraphEdge edge) {
+         return EnglishGrammaticalRelations.DIRECT_OBJECT.equals(edge.getRelation());
+     }
+
+     /** Checks if a given edge holds an indirect object relation */
```

```
+    public static boolean isIobj(SemanticGraphEdge edge) {
+        return EnglishGrammaticalRelations.INDIRECT_OBJECT.equals(edge.getRelation());
+    }
+
+    /** Checks if a given edge holds a negation relation */
+    static boolean isNeg(SemanticGraphEdge edge) {
+        return EnglishGrammaticalRelations.NEGATION_MODIFIER.equals(edge.getRelation());
+    }
+
+    /** Checks if a given edge holds the 'dep' relation */
+    public static boolean isDep(SemanticGraphEdge edge) {
+        return edge.toString().equals("dep");
+    }
+
+    /** Checks if a given edge holds a phrasal verb particle relation */
+    public static boolean isPrt(SemanticGraphEdge edge) {
+        return EnglishGrammaticalRelations.PHRASAL_VERB_PARTICLE.equals(edge.getRelation());
+    }
+
+    /** Checks if a given edge holds an appositional relation */
+    public static boolean isAppos(SemanticGraphEdge edge) {
+        return EnglishGrammaticalRelations.APPOSITIONAL_MODIFIER.equals(edge.getRelation());
+    }
+
+    /** Checks if a given edge holds an purpose clause modifier relation */
+    //  public static boolean isPurpcl(SemanticGraphEdge edge) {
+    //     return EnglishGrammaticalRelations.PURPOSE_CLAUSE_MODIFIER.equals(edge.getRelation());
+    // }
+
+    /** Checks if a given edge holds a xcomp relation */
+    public static boolean isXcomp(SemanticGraphEdge edge) {
+        return EnglishGrammaticalRelations.XCLAUSAL_COMPLEMENT.equals(edge.getRelation());
+    }
+
+    /** Checks if a given edge holds a complementizer relation */
+    // public static boolean isComplm(SemanticGraphEdge edge) {
+    //     return EnglishGrammaticalRelations.COMPLEMENTIZER.equals(edge.getRelation());
+    // }
+
+    /** Checks if a given edge holds an agent relation */
+    public static boolean isAgent(SemanticGraphEdge edge) {
+        return EnglishGrammaticalRelations.AGENT.equals(edge.getRelation());
+    }
+
+    /** Checks if a given edge holds an expletive relation */
+    public static boolean isExpl(SemanticGraphEdge edge) {
+        return EnglishGrammaticalRelations.EXPLETIVE.equals(edge.getRelation());
+    }
+
+    /** Checks if a given edge holds an adjectival complement relation */
+    public static boolean isAcomp(SemanticGraphEdge edge) {
+        return EnglishGrammaticalRelations.ADJECTIVAL_COMPLEMENT.equals(edge.getRelation());
+    }
+
+    /** Checks if a given edge holds a prepositional modifier relation */
+    public static boolean isAnyPrep(SemanticGraphEdge edge) {
+        return EnglishGrammaticalRelations.PREPOSITIONAL_MODIFIER.isAncestor(edge.getRelation());
+    }
+
+    /** Checks if a given edge holds a copular relation */
+    public static boolean isCop(SemanticGraphEdge edge) {
+        return EnglishGrammaticalRelations.COPULA.equals(edge.getRelation());
+    }
+
+    /** Checks if a given edge holds an adverbial clausal relation */
+    public static boolean isAdvcl(SemanticGraphEdge edge) {
+        return EnglishGrammaticalRelations.ADV_CLAUSE_MODIFIER.equals(edge.getRelation());
+    }
+
```

```
+    /** Checks if a given edge holds a relative clause modifier relation */
+    public static boolean isRcmod(SemanticGraphEdge edge) {
+        return EnglishGrammaticalRelations.RELATIVE_CLAUSE_MODIFIER.equals(edge.getRelation());
+    }
+
+    /** Checks if a given edge holds a clausal complement relation */
+    public static boolean isCcomp(SemanticGraphEdge edge) {
+        return EnglishGrammaticalRelations.CLAUSAL_COMPLEMENT.equals(edge.getRelation());
+    }
+
+    /** Checks if a given edge holds an adverbial modifier relation */
+    public static boolean isAdvmod(SemanticGraphEdge edge) {
+        return EnglishGrammaticalRelations.ADVERBIAL_MODIFIER.equals(edge.getRelation());
+    }
+
+    /** Checks if a given edge holds an np adverbial modifier relation */
+    public static boolean isNpadvmod(SemanticGraphEdge edge) {
+        return EnglishGrammaticalRelations.NP_ADVERBIAL_MODIFIER.equals(edge.getRelation());
+    }
+
+    /** Checks if a given edge holds a marker relation */
+    public static boolean isMark(SemanticGraphEdge edge) {
+        return EnglishGrammaticalRelations.MARKER.equals(edge.getRelation());
+    }
+
+    /** Checks if a given edge holds a propositional complement relation */
+    public static boolean isPcomp(SemanticGraphEdge edge) {
+        return EnglishGrammaticalRelations.PREPOSITIONAL_COMPLEMENT.equals(edge.getRelation());
+    }
+
+    /** Checks if a given edge holds a possession modifier relation */
+    public static boolean isPoss(SemanticGraphEdge edge) {
+        return EnglishGrammaticalRelations.POSSESSION_MODIFIER.equals(edge.getRelation());
+    }
+
+    /** Checks if a given edge holds a possessive modifier relation */
+    public static boolean isPosse(SemanticGraphEdge edge) {
+        return EnglishGrammaticalRelations.POSSESSIVE_MODIFIER.equals(edge.getRelation());
+    }
+
+    /** Checks if a given edge holds a participial modifier relation */
+    public static boolean isPartMod(SemanticGraphEdge edge) {
+        return EnglishGrammaticalRelations.VERBAL_MODIFIER.equals(edge.getRelation());
+    }
+
+    /** Checks if a given edge holds a temporal modifier relation */
+    public static boolean isTmod(SemanticGraphEdge edge) {
+        return EnglishGrammaticalRelations.TEMPORAL_MODIFIER.equals(edge.getRelation());
+    }
+
+    /** Checks if a given edge holds a conjunct relation */
+    public static boolean isAnyConj(SemanticGraphEdge edge) {
+        return EnglishGrammaticalRelations.CONJUNCT.isAncestor(edge.getRelation());
+    }
+
+    /** Checks if a given edge holds a preconjunct modifier relation */
+    public static boolean isPreconj(SemanticGraphEdge edge) {
+        return EnglishGrammaticalRelations.PRECONJUNCT.equals(edge.getRelation());
+    }
+
+    /** Checks if a given edge holds a coordination relation */
+    public static boolean isCc(SemanticGraphEdge edge) {
+        return EnglishGrammaticalRelations.COORDINATION.equals(edge.getRelation());
+    }
+
+    /** Checks if a given edge holds an auxiliar modifier relation */
+    public static boolean isAux(SemanticGraphEdge edge) {
+        return EnglishGrammaticalRelations.AUX_MODIFIER.equals(edge.getRelation());
+    }
```

```
+
+     /** Checks if a given edge holds an auxiliar passive modifier relation */
+     public static boolean isAuxPass(SemanticGraphEdge edge) {
+           return EnglishGrammaticalRelations.AUX_PASSIVE_MODIFIER.equals(edge.getRelation());
+     }
+
+     /** Checks if a given edge holds a 'rel' relation */
+     public static boolean isRel(SemanticGraphEdge edge) {
+           return EnglishGrammaticalRelations.RELATIVE.equals(edge.getRelation());
+     }
+
+     /** Checks if a given edge holds a multi word expression relation */
+     public static boolean isMwe(SemanticGraphEdge edge) {
+           return EnglishGrammaticalRelations.MULTI_WORD_EXPRESSION.equals(edge.getRelation());
+     }
+
+     /** Checks if a given edge holds a parataxis relation */
+     public static boolean isParataxis(SemanticGraphEdge edge) {
+           return EnglishGrammaticalRelations.PARATAXIS.equals(edge.getRelation());
+     }
+
+     public static boolean isPassive(SemanticGraphEdge edge) {
+           return EnglishGrammaticalRelations.AUX_PASSIVE_MODIFIER.equals(edge.getRelation());
+     }
+
+
+
+
+//   /** Checks if a given edge holds an infinitival modifier relation */
+     //   public static boolean isInfmod(SemanticGraphEdge edge) {
+//                 return EnglishGrammaticalRelations.INFINITIVAL_MODIFIER.equals(edge.getRelation());
+//          }
+
+     /** Checks if a given edge holds a predeterminer relation */
+     public static boolean isPredet(SemanticGraphEdge edge) {
+           return EnglishGrammaticalRelations.PREDETERMINER.equals(edge.getRelation());
+     }
+
+     /** Removes some edges from the given semantic graph.
+      *
+      * This method traverses the semantic graph starting from the given root. An edge is removed if
+      * (1) its child appears in <code>excludeVertexes</code>, (2) its relation appears in
+      * <code>excludeRelations</code>, or (3) the edge has the root as parent and its relation
+      * appears in <code>excludeRelationsTop</code>. */
+     public static void removeEdges(SemanticGraph graph, IndexedWord root,
+                                    Collection<IndexedWord> excludeVertexes,
+                                    Collection<GrammaticalRelation> excludeRelations,
+                                    Collection<GrammaticalRelation> excludeRelationsTop) {
+           if (!excludeVertexes.contains(root)) {
+                 List<SemanticGraphEdge> edgesToRemove = new ArrayList<SemanticGraphEdge>();
+                 subgraph(graph, root, excludeVertexes, excludeRelations, excludeRelationsTop,
+                       edgesToRemove);
+                 for (SemanticGraphEdge edge : edgesToRemove) {
+                       graph.removeEdge(edge);
+                 }
+           }
+     }
+
+     /** Removes some edges from the given semantic graph.
+      *
+      * This method traverses the semantic graph starting from the given root. An edge is removed if
+      * its child appears in <code>excludeVertexes</code>. */
+     public static void removeEdges(SemanticGraph graph, IndexedWord root,
+                                    Collection<IndexedWord> excludeVertexes) {
+           removeEdges(graph, root, excludeVertexes, Collections.<GrammaticalRelation> emptySet(),
+                       Collections.<GrammaticalRelation> emptySet());
+     }
+
+     /** Removes some edges from the given semantic graph.
```

```
+     *
+     * This method traverses the semantic graph starting from the given root. An edge is removed if
+     * its relation appears in <code>excludeRelations</code> or the edge has the root as parent and
+     * its relation appears in <code>excludeRelationsTop</code>. */
+    public static void removeEdges(SemanticGraph graph, IndexedWord root,
+                                   Collection<GrammaticalRelation> excludeRelations,
+                                   Collection<GrammaticalRelation> excludeRelationsTop) {
+        removeEdges(graph, root, Collections.<IndexedWord> emptySet(), excludeRelations,
+                excludeRelationsTop);
+    }
+
+    /** Implementation for
+     * {@link #removeEdges(SemanticGraph, IndexedWord, Collection, Collection, Collection)} */
+    private static void subgraph(SemanticGraph graph, IndexedWord root,
+                                 Collection<IndexedWord> excludeVertexes,
+                                 Collection<GrammaticalRelation> excludeRelations,
+                                 Collection<GrammaticalRelation> excludeRelationsTop,
+                                 Collection<SemanticGraphEdge> edgesToRemove) {
+        List<SemanticGraphEdge> explored = new ArrayList<SemanticGraphEdge>();
+        subgraph(graph, root, excludeVertexes, excludeRelations, excludeRelationsTop,
+                edgesToRemove, explored);
+    }
+
+    private static void subgraph(SemanticGraph graph, IndexedWord root,
+                                 Collection<IndexedWord> excludeVertexes,
+                                 Collection<GrammaticalRelation> excludeRelations,
+                                 Collection<GrammaticalRelation> excludeRelationsTop,
+                                 Collection<SemanticGraphEdge> edgesToRemove,
+                                 Collection<SemanticGraphEdge> explored) {
+        List<SemanticGraphEdge> edges = graph.getOutEdgesSorted(root);
+        for (SemanticGraphEdge e : edges) {
+            if(explored.contains(e))
+                continue;
+            explored.add(e);
+            IndexedWord child = e.getDependent();
+            if (excludeVertexes.contains(child) || excludeRelations.contains(e.getRelation())
+               || excludeRelationsTop.contains(e.getRelation())) {
+                edgesToRemove.add(graph.getEdge(root, child));
+            } else {
+                subgraph(graph, child, excludeVertexes, excludeRelations,
+                        Collections.<GrammaticalRelation> emptySet(), edgesToRemove, explored);
+            }
+        }
+    }
+
+    /** Disconnects independent clauses by removing the edge representing the coordinating conjunction */
+    public static void disconectClauses(SemanticGraph graph, Constituent constituent) {
+        List<SemanticGraphEdge> outedges = graph
+            .getOutEdgesSorted(((IndexedConstituent) constituent).getRoot());
+        for (int i = 0; i < outedges.size(); i++) {
+            SemanticGraphEdge e = outedges.get(i);
+            if (DpUtils.isAnyConj(e)) {
+                IndexedWord child = e.getDependent();
+                List<SemanticGraphEdge> outNewRoot = graph.getOutEdgesSorted(child);
+                SemanticGraphEdge sub = DpUtils.findFirstOfRelationOrDescendent(outNewRoot,
+                                                    EnglishGrammaticalRelations.SUBJECT);
+                if (sub != null)
+                    graph.removeEdge(e);
+            }
+        }
+    }
+
+    /** Return a set of vertexes to be excluded according to a given collection of grammatical relations */
+    public static Set<IndexedWord> exclude(SemanticGraph semanticGraph,
+                                Collection<GrammaticalRelation> rels, IndexedWord root) {
+        Set<IndexedWord> exclude = new TreeSet<IndexedWord>();
+        List<SemanticGraphEdge> outedges = semanticGraph.getOutEdgesSorted(root);
+        for (SemanticGraphEdge edge : outedges) {
+            if (containsAncestor(rels, edge)) {
```

```
+                     exclude.add(edge.getDependent());
+                 }
+             }
+             return exclude;
+         }
+
+         /** Check if an edge is descendant of any grammatical relation in the given set */
+         private static boolean containsAncestor(Collection<GrammaticalRelation> rels,
+                                                 SemanticGraphEdge edge) {
+             for (GrammaticalRelation rel : rels) {
+                 if (rel.isAncestor(edge.getRelation()))
+                     return true;
+             }
+             return false;
+         }
+
+         /** Correspondence between nodes in Tree and SemanticGraph */
+         public static Tree getNode(IndexedWord word, Tree depTree, SemanticGraph semanticGraph) {
+             int indexSC = semanticGraph.vertexListSorted().indexOf(word);
+             int indexDT = Integer.MAX_VALUE;
+             Tree result = null;
+             List<Tree> descTree = depTree.getLeaves();
+             for(int i = descTree.size() - 1; i >= 0; i--) {
+                 String s = descTree.get(i).toString();
+                 String v = word.value();
+                 if(descTree.get(i).toString().equals(word.value())) {
+                     if((i - indexSC) < indexDT) {
+                         result = descTree.get(i);
+                         indexDT = i - indexSC;
+                     }
+                 }
+             }
+             return result;
+         }
+
+         /** Correspondence between nodes in Tree and SemanticGraph */
+         public static IndexedWord getIndexedWord(Tree word, Integer indexDT, Tree depTree, SemanticGraph semanticGraph) {
+             List<IndexedWord> iwList = semanticGraph.vertexListSorted();
+             List<Tree> descTree = depTree.getLeaves();
+             if(indexDT == null)
+                 indexDT = descTree.indexOf(word);
+             int indexSC = Integer.MAX_VALUE;;
+             IndexedWord result = null;
+             for(int i = iwList.size() - 1; i >= 0; i--) {
+                 if(iwList.get(i).word().equals(word.toString())) {
+                     if(Math.abs((i - indexDT)) < indexSC) {
+                         result = iwList.get(i);
+                         indexSC = Math.abs(i - indexDT);
+                     }
+                 }
+             }
+             return result;
+         }
+
+         public static boolean findRelClause(List<SemanticGraphEdge> outEdgesSorted) {
+             for(SemanticGraphEdge edge: outEdgesSorted) {
+                 if(edge.getDependent().tag().charAt(0) == 'W')
+                     return true;
+             }
+             return false;
+         }
+
+         public static int getindex(IndexedWord iw, Tree tree, SemanticGraph semanticGraph) {
+             Tree node = getNode(iw, tree, semanticGraph);
+             return tree.getLeaves().indexOf(node);
+         }
+
+         public static IndexedWord getIndexedWord(int index, Tree tree, SemanticGraph semanticGraph) {
+             Tree node = tree.getLeaves().get(index);
```

```
+            return getIndexedWord(node, index, tree, semanticGraph);
+      }
+
+
+}
diff --git a/src/clausie/IndexedConstituent.java b/src/clausie/IndexedConstituent.java
new file mode 100644
index 0000000..5e58d3a
--- /dev/null
+++ b/src/clausie/IndexedConstituent.java
@@ -0,0 +1,167 @@
+package clausie;
+
+import java.util.List;
+import java.util.Set;
+import java.util.TreeSet;
+
+import edu.stanford.nlp.ling.IndexedWord;
+import edu.stanford.nlp.semgraph.SemanticGraph;
+import edu.stanford.nlp.semgraph.SemanticGraphEdge;
+import edu.stanford.nlp.semgraph.SemanticGraphFactory;
+
+/** A constituent of a clause described by a {@link SemanticGraph}.
+ *
+ * Each constituent has a root vertex. The root together with its the descendants form the
+ * constituent. In some cases, additional vertexes need to be included or excluded;
+ * these vertexes are also recorded within this class.
+ *
+ * Note that the {@link SemanticGraph} may or may not match the graph of the input sentences or the
+ * other constituents of the same clause. For example, the semantic graphs are modified when
+ * processing of coordinating conjunctions.
+ *
+ * @date $LastChangedDate: 2013-06-09 15:16:12 +0200 (Sun, 09 Jun 2013) $
+ * @version $LastChangedRevision: 799 $ */
+public class IndexedConstituent extends Constituent {
+
+      // -- member variables ------------------------------------------------------------------------
+
+      /** Semantic graph for this sentence */
+      protected static SemanticGraph sentSemanticGraph;
+
+      /** Semantic graph for this constituent */
+      protected SemanticGraph semanticGraph;
+
+      /** The root vertex of this constituent in {@link #semanticGraph}. This vertex and all its
+       * descendants are part of the constituent (unless they appear in {@link #excludedVertexes}). */
+      protected IndexedWord root;
+
+      /** Additional root vertexes that form this constituent. These vertexes and all their descendants
+       * are part of the constituent (unless they appear in {@link #excludedVertexes}). */
+      protected Set<IndexedWord> additionalVertexes;
+
+      /** Vertexes that are excluded from this constituent. All descendants are excluded as well
+       * (unless they appear in {@link #root} or {@link additionalRoots}). */
+      protected Set<IndexedWord> excludedVertexes;
+
+      // -- construction ----------------------------------------------------------------------------
+
+      protected IndexedConstituent() {
+      }
+
+      /** Constructs a new indexed constituent.
+       *
+       * @param semanticGraph Semantic graph for this constituent ({@see #semanticGraph})
+       * @param root The root vertex of this constituent ({@see {@link #root})
+       * @param additionalVertexes Additional root vertexes that form this constituent ({@see
+       *          {@link #additionalVertexes})
+       * @param excludedVertexes Vertexes that are excluded from this constituent ({@see
+       *          {@link #excludedVertexes})
```

```
+     * @param type type of this constituent */
+    public IndexedConstituent(SemanticGraph semanticGraph, IndexedWord root,
+                              Set<IndexedWord> additionalVertexes, Set<IndexedWord> excludedVertexes, Type type) {
+        super(type);
+        this.semanticGraph = semanticGraph;
+        this.root = root;
+        this.additionalVertexes = new TreeSet<IndexedWord>(additionalVertexes);
+        this.excludedVertexes = new TreeSet<IndexedWord>(excludedVertexes);
+    }
+
+    /** Constructs a simple indexed constituent without additional additional or excluded vertexes.
+     *
+     * @param semanticGraph Semantic graph for this constituent ({@see #semanticGraph})
+     * @param root The root vertex of this constituent ({@see {@link #root})
+     * @param type type of this constituent */
+    public IndexedConstituent(SemanticGraph semanticGraph, IndexedWord root, Type type) {
+        this(semanticGraph, root, new TreeSet<IndexedWord>(), new TreeSet<IndexedWord>(), type);
+    }
+
+    /** Creates a deep copy of this indexed constituent. */
+    @Override
+    public IndexedConstituent clone() {
+        IndexedConstituent clone = new IndexedConstituent();
+        clone.type = type;
+        //clone.semanticGraph = new SemanticGraph(semanticGraph);
+        clone.semanticGraph = SemanticGraphFactory.duplicateKeepNodes(semanticGraph);
+        clone.root = this.root;
+        clone.additionalVertexes = new TreeSet<IndexedWord>(this.additionalVertexes);
+        clone.excludedVertexes = new TreeSet<IndexedWord>(this.excludedVertexes);
+        return clone;
+    }
+
+    // -- getters/setters -------------------------------------------------------------------
+
+    /** Returns the semantic graph for this constituent ({@see #semanticGraph}). */
+    public SemanticGraph getSemanticGraph() {
+        return semanticGraph;
+    }
+
+    /** Returns the semantic graph for this sentence ({@see #sentSemanticGraph}). */
+    public SemanticGraph getSentSemanticGraph() {
+        return sentSemanticGraph;
+    }
+
+    /** Sets the semantic graph for this constituent ({@see #semanticGraph}). */
+    public void setSemanticGraph(SemanticGraph newSemanticGraph) {
+        semanticGraph = newSemanticGraph;
+    }
+
+    /** Returns the root vertex of this constituent ({@see {@link #root}). */
+    public IndexedWord getRoot() {
+        return root;
+    }
+
+    /** Sets the root vertex of this constituent ({@see {@link #root}). */
+    public void setRoot(IndexedWord newRoot) {
+        root = newRoot;
+    }
+
+    /** Returns additional root vertexes that form this constituent ({@see
+     * {@link #additionalVertexes}). */
+    public Set<IndexedWord> getAdditionalVertexes() {
+        return additionalVertexes;
+    }
+
+    /** Returns vertexes that are excluded from this constituent ({@see {@link #excludedVertexes}). */
+    public Set<IndexedWord> getExcludedVertexes() {
+        return excludedVertexes;
+    }
```

```
+
+	/** Checks whether this constituent is a prepositional phrase (i.e., starts with a preposition). */
+	public boolean isPrepositionalPhrase() { //This is a mess, find other way of fixing. This is purelly heuristic. It needs to know the semantic graph for the sentence after this is fixed the member variable sentSemanticGraph can be removed
+		List<IndexedWord> parents = semanticGraph.getParentList(root); //This is not the cleanest way semantics messed up. specially with the rel we cannot just check if the head is a preposition (return root.tag().equals("IN")) because the parser some times includes a preposition in the verbal phrase "He is about to win"
+		for(IndexedWord parent: parents) {
+			SemanticGraphEdge edge = semanticGraph.getEdge(parent, root);
+			if(DpUtils.isRel(edge))
+				return true;
+			if(DpUtils.isAnyPrep(edge)) {
+				List<IndexedWord> ancestors = semanticGraph.getParentList(parent);
+				for(IndexedWord ancestor: ancestors) {
+					SemanticGraphEdge ed = semanticGraph.getEdge(ancestor, parent);
+					if(DpUtils.isRcmod(ed))
+						return true;
+				}
+			}
+		}
+		return false;
+		//return root.tag().equals("IN");
+	}
+
+	// -- utility methods ----------------------------------------------------------------
+
+	/** Returns a textual representation of the root word of this constituent. */
+	public String rootString() {
+		return root.word();
+	}
+
+	/** Returns a copy of the semantic graph of this constituent in which all edges (from any
+	 * included vertex) to excluded vertexes have been removed. Useful for proposition generation. */
+	public SemanticGraph createReducedSemanticGraph() {
+		//SemanticGraph result = new SemanticGraph(semanticGraph);
+		 SemanticGraph result = SemanticGraphFactory.duplicateKeepNodes(semanticGraph);
+		DpUtils.removeEdges(result,  root,  excludedVertexes);
+		for (IndexedWord v : additionalVertexes) {
+		  DpUtils.removeEdges(result,  v,  excludedVertexes);
+		}
+		return result;
+	}
+}
diff --git a/src/clausie/JavaUtils.java b/src/clausie/JavaUtils.java
new file mode 100644
index 0000000..e84521f
--- /dev/null
+++ b/src/clausie/JavaUtils.java
@@ -0,0 +1,32 @@
+package clausie;
+import java.util.*;
+
+
+public class JavaUtils {
+
+
+	public static class MapUtil
+	{
+		public static <K, V extends Comparable<? super V>> Map<K, V>
+			sortByValue( Map<K, V> map )
+		    {
+			List<Map.Entry<K, V>> list =
+			  new LinkedList<Map.Entry<K, V>>( map.entrySet() );
+			Collections.sort( list, new Comparator<Map.Entry<K, V>>()
+			{
+			  public int compare( Map.Entry<K, V> o1, Map.Entry<K, V> o2 )
+			  {
+			    return (o1.getValue()).compareTo( o2.getValue() );
+			  }
```

```
+                    } );
+
+                    Map<K, V> result = new LinkedHashMap<K, V>();
+                    for (Map.Entry<K, V> entry : list)
+                    {
+                       result.put( entry.getKey(), entry.getValue() );
+                    }
+                    return result;
+                 }
+              }
+
+}
diff --git a/src/clausie/NounPhraseIndexedConstituent.java b/src/clausie/NounPhraseIndexedConstituent.java
new file mode 100644
index 0000000..44b78fa
--- /dev/null
+++ b/src/clausie/NounPhraseIndexedConstituent.java
@@ -0,0 +1,70 @@
+package clausie;
+
+import java.util.List;
+import java.util.Set;
+import edu.stanford.nlp.ling.IndexedWord;
+import edu.stanford.nlp.semgraph.SemanticGraph;
+import edu.stanford.nlp.semgraph.SemanticGraphEdge;
+
+public class NounPhraseIndexedConstituent extends StructuredConstituent{
+
+
+    public NounPhraseIndexedConstituent(SemanticGraph semanticGraph, IndexedWord root,
+                                Type type) {
+        super(semanticGraph, root, type);
+    }
+    /** Constructs a new indexed constituent for any noun-phrase relation.
+     *
+     * @param semanticGraph Semantic graph for this constituent ({@see #semanticGraph})
+     * @param root The root vertex of this constituent ({@see {@link #root})
+     * @param additionalVertexes Additional root vertexes that form this constituent ({@see
+     *          {@link #additionalVertexes})
+     * @param excludedVertexes Vertexes that are excluded from this constituent ({@see
+     *          {@link #excludedVertexes})
+     * @param type type of this constituent
+     * * @param clauses derived from this constituent*/
+    public NounPhraseIndexedConstituent(SemanticGraph semanticGraph, IndexedWord root,
+                                Set<IndexedWord> additionalVertexes,
+                                Set<IndexedWord> excludedVertexes, Type type) {
+        super(semanticGraph, root, additionalVertexes, excludedVertexes, type);
+    }
+
+    public NounPhraseIndexedConstituent build() {
+        List<SemanticGraphEdge> edges = semanticGraph.outgoingEdgeList(root);
+        for(SemanticGraphEdge edge: edges) {
+            if(DpUtils.isAnyPrep(edge)) {
+                getConstituents().add(new PrepositionalPhraseIndexedConstituent(semanticGraph, root, null).build());
+                getPrepPhrases().add(getConstituents().size() - 1);
+            } else if(DpUtils.isAdvmod(edge)) {
+                getConstituents().add(new IndexedConstituent(semanticGraph, root, null));
+                advmods.add(getConstituents().size() - 1);
+            } else if(DpUtils.isAMod(edge)) {
+                getConstituents().add(new IndexedConstituent(semanticGraph, root, null));
+                getAmods().add(getConstituents().size() - 1);
+            } else if(DpUtils.isNN(edge)) {
+                getConstituents().add(new IndexedConstituent(semanticGraph, root, null));
+                getNn().add(getConstituents().size() - 1);
+            } else if(DpUtils.isAppos(edge)) {
+                getConstituents().add(new NounPhraseIndexedConstituent(semanticGraph, root, null).build());
+                appos.add(getConstituents().size() - 1);
+            } else if(DpUtils.isAnyConj(edge)) {
+                //Check it should be other noun phrase
```

```
+                        getConstituents().add(new IndexedConstituent(semanticGraph, root, null));
+                        conj.add(getConstituents().size() - 1);
+                        //} else if(DpUtils.isInfmod(edge)) {
+                        //  getConstituents().add(new IndexedConstituent(semanticGraph, root, null));
+                        //  infmod.add(getConstituents().size() - 1);
+                        //} else if(DpUtils.isPartMod(edge)) {
+                        //  getConstituents().add(new IndexedConstituent(semanticGraph, root, null));
+                        //  partmod.add(getConstituents().size() - 1);
+                    } else if(DpUtils.isRcmod(edge)) {
+                        //This is another clause check
+                        getConstituents().add(new IndexedConstituent(semanticGraph, root, null));
+                        rcmod.add(getConstituents().size() - 1);
+                    }
+
+                            }
+                            return this;
+                }
+
+}
diff --git a/src/clausie/Options.java b/src/clausie/Options.java
new file mode 100644
index 0000000..366cc0b
--- /dev/null
+++ b/src/clausie/Options.java
@@ -0,0 +1,233 @@
+package clausie;
+
+import java.io.File;
+import java.io.FileInputStream;
+import java.io.FileNotFoundException;
+import java.io.IOException;
+import java.io.InputStream;
+import java.io.OutputStream;
+import java.io.PrintStream;
+import java.net.URL;
+import java.util.Arrays;
+import java.util.Iterator;
+import java.util.Properties;
+import java.util.Set;
+
+import edu.stanford.nlp.ling.IndexedWord;
+
+/** Options handles the ClausIe settings which should be loaded out of a configuration file.
+ *
+ * @date $LastChangedDate: 2013-04-24 11:35:23 +0200 (Wed, 24 Apr 2013) $
+ * @version $LastChangedRevision: 739 $ */
+public class Options {
+    // informatin
+    public Dictionary dictCopular;
+    public Dictionary dictExtCopular;
+    public Dictionary dictNotExtCopular;
+    public Dictionary dictComplexTransitive;
+    public Dictionary dictAdverbsConj;
+    public Dictionary dictAdverbsIgnore;
+    public Dictionary dictAdverbsInclude;
+    public boolean conservativeSVA;
+    public boolean conservativeSVOA;
+
+    /**
+     * Process coordinating conjunctions with common components. All other verbal coordinating
+     * conjunctions will always been processed.
+     *
+     * Example: some sentence
+     * Option on: ...
+     * Option off:
+     *
+     * defaulot value
+     *
+     * Example sentance that is not affected
```

```
+      */
+     public boolean processCcAllVerbs;
+     public boolean processCcNonVerbs;
+     public boolean processAppositions;
+     public boolean processPossessives;
+     public boolean processPartmods;
+//          public boolean processPassive = false; // NOT SUPPORTED FOR NOW (collapsed semantic graph needed but less stable)
+     //add for possesive
+
+     // representation
+     public boolean nary;
+     public int minOptionalArgs; // only when nary=false
+     public int maxOptionalArgs; // only when nary=false
+     public boolean lemmatize;
+     public String appositionVerb;
+     public String possessiveVerb;
+
+     //WSD
+     public int appositionVerbSynset;
+     public int possessiveVerbSynset;
+     public int existentialVerbSynset;
+
+
+     //helpds
+
+     /**Constructs the set of options out of a conf file (clausie.conf)*/
+     public Options() {
+          try {
+               URL t = getClass().getResource("resources/clausie.conf");
+               InputStream in = t.openStream();
+               setOptions(in);
+               in.close();
+          } catch (IOException e) {
+               // should not happen
+               throw new RuntimeException(e);
+          }
+     }
+
+     /**Constructs the set of options out of a conf file (fileOrResourceName)*/
+     public Options(String fileOrResourceName) throws IOException {
+          InputStream in = openFileOrResource(fileOrResourceName);
+          setOptions(in);
+          in.close();
+     }
+
+     private InputStream openFileOrResource(String name) throws IOException {
+          try {
+               File file = new File(name);
+               return new FileInputStream(file);
+          } catch (FileNotFoundException e) {
+          }
+          URL url = getClass().getResource(name);
+          if (url == null) {
+               throw new IOException("File or resource '" + name + "' not found.");
+          }
+          return url.openStream();
+     }
+
+     /** Load options from the configuration file*/
+     public void setOptions(InputStream optionsStream) throws IOException {
+          Properties prop = new Properties();
+          prop.load(optionsStream);
+
+          // load the required options
+          conservativeSVA = Boolean.parseBoolean(getProperty(prop, "conservativeSVA"));
+          conservativeSVOA = Boolean.parseBoolean(getProperty(prop, "conservativeSVOA"));
+          processCcAllVerbs = Boolean.parseBoolean(getProperty(prop, "processCcAllVerbs"));
+          processCcNonVerbs = Boolean.parseBoolean(getProperty(prop, "processCcNonVerbs"));
+          processAppositions = Boolean.parseBoolean(getProperty(prop, "processAppositions"));
```

```
+            appositionVerb = getProperty(prop, "appositionVerb");
+            processPossessives = Boolean.parseBoolean(getProperty(prop, "processPossessives"));
+            possessiveVerb = getProperty(prop, "possessiveVerb");
+            processPartmods = Boolean.parseBoolean(getProperty(prop, "processPartmods"));
+            lemmatize = Boolean.parseBoolean(getProperty(prop, "lemmatize"));
+            nary = Boolean.parseBoolean(getProperty(prop, "nary"));
+            minOptionalArgs = Integer.parseInt(getProperty(prop, "minOptionalArgs"));
+            maxOptionalArgs = Integer.parseInt(getProperty(prop, "maxOptionalArgs"));
+            appositionVerbSynset = Integer.parseInt(getProperty(prop, "appositionVerbSynset"));
+            possessiveVerbSynset = Integer.parseInt(getProperty(prop, "possessiveVerbSynset"));
+            existentialVerbSynset = Integer.parseInt(getProperty(prop, "existentialVerbSynset"));
+
+
+//                          // get dictionaries
+            dictCopular = getDictionary(prop, "dictCopular");
+            dictExtCopular = getDictionary(prop, "dictExtCopular");
+            dictNotExtCopular = getDictionary(prop, "dictNotExtCopular");
+            dictComplexTransitive = getDictionary(prop, "dictComplexTransitive");
+            dictAdverbsConj = getDictionary(prop, "dictAdverbsConj");
+            dictAdverbsIgnore = getDictionary(prop, "dictAdverbsIgnore");
+            dictAdverbsInclude = getDictionary(prop, "dictAdverbsInclude");
+
+            // check for unused properties
+            if (!prop.isEmpty()) {
+                System.err.println( "Unknown option(s): "
+                                    + Arrays.toString( prop.keySet().toArray() ));
+            }
+        }
+
+        /** Returns a required option (key) */
+        private String getProperty(Properties prop, String key) throws IOException {
+            String result = prop.getProperty(key);
+            if (result == null) {
+                throw new IOException("Missing option: " + key);
+            }
+            prop.remove(key);
+            return result;
+        }
+
+        /**Loads a dictionary (key) */
+        private Dictionary getDictionary(Properties prop, String key) throws IOException {
+            String name = getProperty(prop, key);
+            InputStream in = openFileOrResource(name);
+            Dictionary dict = new Dictionary();
+            dict.load(in);
+            in.close();
+            return dict;
+        }
+
+        /**Checks if the copular dictionary contains a given word*/
+        public boolean isCop(IndexedWord word) {
+            return dictCopular.contains(word);
+        }
+
+        /**Checks if the extended copular dictionary contains a given word*/
+        public boolean isExtCop(IndexedWord word) {
+            return dictExtCopular.contains(word);
+        }
+
+        /**Checks if the non-extended copular dictionary contains a given word*/
+        public boolean isNotExtCop(IndexedWord word) {
+            return dictNotExtCopular.contains(word);
+        }
+
+        /**Checks if the complex transitive dictionary contains a given word*/
+        public boolean isComTran(IndexedWord word) {
+            return dictComplexTransitive.contains(word);
+        }
+
```

```
+	/**Returns a string with some initial words of a given dictionary*/
+	private String someWords(Set<String> dict) {
+		if (dict.isEmpty()) return "";
+		StringBuffer result = new StringBuffer();
+		Iterator<String> it = dict.iterator();
+		String sep = "";
+		result.append(" (");
+		for(int i=0; i<3 && it.hasNext(); i++) {
+			result.append(sep);
+			result.append(it.next());
+			sep = ", ";
+		}
+		if (it.hasNext()) result.append(", ...");
+		result.append(")");
+		return result.toString();
+	}
+
+	public void print(OutputStream out) {
+		print(out, "");
+	}
+
+	/**Print settings*/
+	public void print(OutputStream out, String prefix) {
+		PrintStream pout = new PrintStream(out);
+
+		pout.println(prefix + "CLAUSE DETECTION");
+		pout.println(prefix + "  Dict. copular       : " + dictCopular.size() + someWords(dictCopular.words));
+		pout.println(prefix + "  Dict. ext-copular   : " + dictExtCopular.size() + someWords(dictExtCopular.words));
+		pout.println(prefix + "  Dict. not ext.-cop. : " + dictNotExtCopular.size() + someWords(dictNotExtCopular.words));
+		pout.println(prefix + "  Dict. complex trans. : " + dictComplexTransitive.size() + someWords(dictComplexTransitive.words));
+		pout.println(prefix + "  Dict. ignored adverb : " + dictAdverbsIgnore.size() + someWords(dictAdverbsIgnore.words));
+		pout.println(prefix + "  Dict. included adverb: " + dictAdverbsInclude.size() + someWords(dictAdverbsInclude.words));
+		pout.println(prefix + "  Dict. conj adverbs   : " + dictAdverbsConj.size() + someWords(dictAdverbsConj.words));
+		pout.println(prefix + "  Conservative SVA    : " + conservativeSVA);
+		pout.println(prefix + "  Conservative SVOA    : " + conservativeSVOA);
+		pout.println(prefix + "  Process all verb CCs : " + processCcAllVerbs);
+		pout.println(prefix + "  Process non-verb CCs : " + processCcNonVerbs);
+		pout.println(prefix + "  Process appositions  : " + processAppositions);
+		pout.println(prefix + "  Process possessives  : " + processPossessives);
+		pout.println(prefix + "  Process partmods     : " + processPartmods);
+
+		pout.println(prefix + "");
+		pout.println(prefix + "REPRESENTATION");
+			pout.println(prefix + "  n-ary propositions  : " + nary);
+			pout.println(prefix + "  Min. opt. args      : " + minOptionalArgs);
+			pout.println(prefix + "  Max. opt. args      : " + maxOptionalArgs);
+			pout.println(prefix + "  Lemmatize           : " + lemmatize);
+			pout.println(prefix + "  Appositions verb    : \"" + appositionVerb + "\"");
+			pout.println(prefix + "  Possessive verb     : \"" + possessiveVerb + "\"");
+		}
+}
diff --git a/src/clausie/PrepositionalPhraseIndexedConstituent.java b/src/clausie/PrepositionalPhraseIndexedConstituent.java
new file mode 100644
index 0000000..06ce2a0
--- /dev/null
+++ b/src/clausie/PrepositionalPhraseIndexedConstituent.java
@@ -0,0 +1,87 @@
+package clausie;
+
+import java.util.List;
+import java.util.Set;
+
+import edu.stanford.nlp.ling.IndexedWord;
+import edu.stanford.nlp.semgraph.SemanticGraph;
+import edu.stanford.nlp.semgraph.SemanticGraphEdge;
+
+public class PrepositionalPhraseIndexedConstituent extends StructuredConstituent {
+
+	private IndexedWord pobj;
```

```
+
+     public PrepositionalPhraseIndexedConstituent(SemanticGraph semanticGraph, IndexedWord root,
+                                     Type type) {
+          super(semanticGraph, root, type);
+     }
+     /** Constructs a new indexed constituent for any noun-phrase relation.
+      *
+      * @param semanticGraph Semantic graph for this constituent ({@see #semanticGraph})
+      * @param root The root vertex of this constituent ({@see {@link #root})
+      * @param additionalVertexes Additional root vertexes that form this constituent ({@see
+      *         {@link #additionalVertexes})
+      * @param excludedVertexes Vertexes that are excluded from this constituent ({@see
+      *         {@link #excludedVertexes})
+      * @param type type of this constituent
+      * * @param clauses derived from this constituent*/
+     public PrepositionalPhraseIndexedConstituent(SemanticGraph semanticGraph, IndexedWord root,
+                                     Set<IndexedWord> additionalVertexes,
+                                     Set<IndexedWord> excludedVertexes, Type type) {
+          super(semanticGraph, root, additionalVertexes, excludedVertexes, type);
+     }
+
+     public PrepositionalPhraseIndexedConstituent build() {
+          if(!semanticGraph.getChildList(root).isEmpty())
+               setPobj(semanticGraph.getChildList(root).get(0));
+          else
+               return this;
+          List<SemanticGraphEdge> edges = semanticGraph.outgoingEdgeList(getPobj());
+          for(SemanticGraphEdge edge: edges) {
+               if(DpUtils.isAnyPrep(edge)) {
+                    getConstituents().add(new PrepositionalPhraseIndexedConstituent(semanticGraph, edge.getDependent(),
null).build());
+                    getPrepPhrases().add(getConstituents().size() - 1);
+               } else if(DpUtils.isAdvmod(edge)) {
+                    getConstituents().add(new IndexedConstituent(semanticGraph, edge.getDependent(), null));
+                    advmods.add(getConstituents().size() - 1);
+               } else if(DpUtils.isAMod(edge)) {
+                    getConstituents().add(new IndexedConstituent(semanticGraph, edge.getDependent(), null));
+                    getAmods().add(getConstituents().size() - 1);
+               } else if(DpUtils.isNN(edge)) {
+                    getConstituents().add(new IndexedConstituent(semanticGraph, edge.getDependent(), null));
+                    getNn().add(getConstituents().size() - 1);
+               } else if(DpUtils.isAppos(edge)) {
+                    getConstituents().add(new IndexedConstituent(semanticGraph, edge.getDependent(), null));
+                    appos.add(getConstituents().size() - 1);
+               } else if(DpUtils.isAnyConj(edge)) {
+                    getConstituents().add(new IndexedConstituent(semanticGraph, edge.getDependent(), null));
+                    conj.add(getConstituents().size() - 1);
+                    //} else if(DpUtils.isInfmod(edge)) {
+                    //  getConstituents().add(new IndexedConstituent(semanticGraph, edge.getDependent(), null));
+                    //  infmod.add(getConstituents().size() - 1);
+                    //} else if(DpUtils.isPartMod(edge)) {
+                    //  getConstituents().add(new IndexedConstituent(semanticGraph, edge.getDependent(), null));
+                    //  partmod.add(getConstituents().size() - 1);
+               } else if(DpUtils.isRcmod(edge)) {
+                    getConstituents().add(new IndexedConstituent(semanticGraph, edge.getDependent(), null));
+                    rcmod.add(getConstituents().size() - 1);
+               } else if(DpUtils.isAdvcl(edge)) {
+                    //anothe clause check
+                    getConstituents().add(new IndexedConstituent(semanticGraph, edge.getDependent(), null));
+                    advcl.add(getConstituents().size() - 1);
+               }
+
+                    }
+                    return this;
+           }
+
+
+           public IndexedWord getPobj() {
+                    return pobj;
```

```
+		}
+
+
+		public void setPobj(IndexedWord pobj) {
+			this.pobj = pobj;
+		}
+}
diff --git a/src/clausie/ProcessConjunctions.java b/src/clausie/ProcessConjunctions.java
new file mode 100644
index 0000000..88106d7
--- /dev/null
+++ b/src/clausie/ProcessConjunctions.java
@@ -0,0 +1,598 @@
+package clausie;
+
+import java.util.ArrayList;
+import java.util.Collection;
+import java.util.List;
+import java.util.Set;
+
+import clausie.Constituent.Type;
+
+import edu.stanford.nlp.ling.HasWord;
+import edu.stanford.nlp.ling.IndexedWord;
+import edu.stanford.nlp.semgraph.SemanticGraph;
+import edu.stanford.nlp.semgraph.SemanticGraphEdge;
+import edu.stanford.nlp.trees.EnglishGrammaticalRelations;
+import edu.stanford.nlp.trees.GrammaticalRelation;
+import edu.stanford.nlp.trees.Tree;
+
+/** This is a provisory implementation of the processing of coordinating conjunctions.
+ *
+ * Coordinating conjunctions are still a difficult issue for the parser and therefore
+ * the source of a significant loss in precision by ClausIE.
+ *
+ * Code is not clean or optimally efficient. More work needs to be done in how to handle CCs.
+ *
+ * @date $  $
+ * @version $ $ */
+public class ProcessConjunctions {
+
+
+	/** Process CCs of a given constituent */
+	public static List<Constituent> processCC(Tree depTree,
+			Constituent constituent, boolean processVerb, boolean completeProcess, Integer levels, IndexedWord
clauseRoot, List<IndexedWord> roots) {
+		return generateConstituents(depTree, (IndexedConstituent) constituent, processVerb, completeProcess, levels, clauseRoot,
roots);
+	}
+
+	/** Generates a set of constituents from a CC for a given constituent
+	 * @param clauseRoot
+	 * @param roots */
+	private static List<Constituent> generateConstituents(Tree depTree,
+			IndexedConstituent constituent, boolean processVerb, boolean completeProcess, Integer levels,
IndexedWord clauseRoot, List<IndexedWord> roots) {
+		IndexedConstituent copy = constituent.clone();
+		//copy.setSemanticGraph( copy.createReducedSemanticGraph() );
+		copy.setSemanticGraph( copy.getSemanticGraph() );
+		List<Constituent> result = new ArrayList<Constituent>();
+		result.add(copy);
+		generateConstituents(copy.getSemanticGraph(), depTree, copy, copy.getRoot(),
+				result, true, processVerb, completeProcess, levels, clauseRoot, roots);
+		removeUnnecessary(result); //To assure consistency, vertex cannot be eliminated because it affects indices so we eliminate
edges
+		return result;
+
+	}
+
```

```
+    public static void removeUnnecessary(List<Constituent> result) {
+        for(Constituent c: result) {
+            SemanticGraph semanticGraph = ((IndexedConstituent) c).getSemanticGraph();
+            IndexedWord root = ((IndexedConstituent) c).getRoot();
+            removeUnnecessary(semanticGraph, root);
+        }
+    }
+
+    public static void removeUnnecessary(SemanticGraph semanticGraph, IndexedWord root) {
+        List<SemanticGraphEdge> edges = semanticGraph.edgeListSorted();
+        Set<IndexedWord> descendants = semanticGraph.descendants(root);
+        for(int i =0; i < edges.size(); i++) {
+            if(!descendants.contains(edges.get(i).getDependent()) || !descendants.contains(edges.get(i).getGovernor())) {
+                semanticGraph.removeEdge(edges.get(i));
+                edges = semanticGraph.edgeListSorted();
+                i--;
+            }
+        }
+    }
+
+    // Process CCs by exploring the graph from one constituent and generating more constituents as
+    // it encounters ccs
+    private static void generateConstituents(SemanticGraph semanticGraph, Tree depTree,
+                                IndexedConstituent constituent, IndexedWord root, List<Constituent> constituents,
+                                boolean firstLevel, boolean processVerb, boolean completeProcess, Integer levels
+                                ,IndexedWord clauseRoot, List<IndexedWord> roots) {
+
+        if(clauseRoot != null && roots != null && !root.equals(clauseRoot) && roots.contains(root))
+            return;
+
+
+        if(levels != null && levels < 1)
+            return;
+
+        List<SemanticGraphEdge> outedges = semanticGraph.getOutEdgesSorted(root);
+        List<SemanticGraphEdge> conjunct = DpUtils.getEdges(outedges,
+                                        EnglishGrammaticalRelations.COORDINATION);
+
+        Boolean processCC = true;
+        SemanticGraphEdge predet = null;
+
+        //to avoid processing under certain circunstances must be design properly when final setup is decided
+        if (conjunct != null && !conjunct.isEmpty()) {
+            for(SemanticGraphEdge edge: outedges) {
+                if(edge.getDependent().lemma().equals("between")) {
+                    processCC = false;
+                    break;
+                }
+            }
+            SemanticGraphEdge con = DpUtils.findFirstOfRelation(outedges,
+                                            EnglishGrammaticalRelations.QUANTIFIER_MODIFIER);
+            if (con != null && con.getDependent().lemma().equals("between"))
+                processCC = false;
+            List<SemanticGraphEdge> inedg = semanticGraph
+                .getIncomingEdgesSorted(root);
+            SemanticGraphEdge pobj = DpUtils.findFirstOfRelation(inedg,
+                                             EnglishGrammaticalRelations.PREPOSITIONAL_OBJECT);
+            // this wont work with collapsed dependencies
+            if (pobj != null && pobj.getGovernor().lemma().equals("between"))
+                processCC = false;
+            Collection<IndexedWord> sibs = null;
+            try {
+                sibs = semanticGraph.getSiblings(root);
+            } catch(Exception e) {
+                System.out.print(root.toString());
+            }
+            for (IndexedWord sib : sibs) {
+                List<SemanticGraphEdge> insib = semanticGraph
+                    .getIncomingEdgesSorted(sib);
```

```
+                       predet = DpUtils.findFirstOfRelation(insib,
+                                       EnglishGrammaticalRelations.PREDETERMINER);
+                       if (predet == null)
+                           predet = DpUtils.findFirstOfRelation(insib,
+                                           EnglishGrammaticalRelations.DETERMINER);
+                       if (predet != null)
+                           break;
+                   }
+               }
+
+
+               for (SemanticGraphEdge edge : outedges) {
+                   // ClausIE requires this uncommented                   if (DpUtils.isParataxis(edge) || DpUtils.isRcmod(edge) ||
DpUtils.isAppos(edge)  && !processVerb) //||(DpUtils.isDep(edge) && constituent.type.equals(Type.VERB)
+                   //                      continue;//to avoid processing relative clauses and appositions which are included as an independent
clause in the clauses list of the sentence, also no dep in verbs are processed. To reproduce the results of the paper comment this line and
eliminate the duplicate propositions that may be generated.
+                   if (DpUtils.isAnyConj(edge) && processCC) {
+
+
+                       if(roots != null&& roots.contains(edge.getDependent()))
+                           continue;;
+
+
+                       boolean cont = false;
+                       for(SemanticGraphEdge c : conjunct) {
+                           if(c.getDependent().lemma().equals("&") && nextToVerb(depTree, semanticGraph, root, edge.getDependent(),
c.getDependent())) {
+                               //nextToVerb(depTree, root.index(), edge.getDependent().index(), c.getDependent().index())) {
+                               cont = true;
+                               break;
+                           }
+                       }
+
+                       if(cont)
+                           continue;
+
+                       IndexedWord newRoot = edge.getDependent();
+                       if(predet != null && predet.getDependent().lemma().equals("both"))
+                           constituent.getExcludedVertexes().add(predet.getDependent());
+
+
+
+                       List<IndexedConstituent> newConstituents = new ArrayList<IndexedConstituent>();
+                       for(Constituent c : constituents) {
+
if(!((IndexedConstituent)c).getSemanticGraph().descendants(((IndexedConstituent)c).getSemanticGraph().getFirstRoot()).contains(root
))
+                               continue;
+
+
+                           if(!containsDescendant(depTree, ((IndexedConstituent)c).getSemanticGraph(), semanticGraph, root, newRoot) &&
shareAllAncestors(semanticGraph, ((IndexedConstituent)c).getSemanticGraph(), root))
+                               continue;
+
+
+
+                           IndexedConstituent newConstituent = ((IndexedConstituent) c).clone();
+                           if (firstLevel) {
+                               newConstituent.setRoot(newRoot);
+                           }
+                           newConstituents.add(newConstituent);
+                       }
+
+                       boolean isTreeRoot = false;
+                       //To process verbs if the main conj is the root of the graph
+                       if(!semanticGraph.getRoots().contains(root)) {
+                           // Assign all the parents to the conjoint
+                           for(Constituent c: newConstituents) {
```

```
+                        SemanticGraph newSemanticGraph = ((IndexedConstituent) c).getSemanticGraph();
+                        Collection<IndexedWord> parents = newSemanticGraph.getParents(root);
+                        for (IndexedWord parent : parents) {
+                            GrammaticalRelation reln = newSemanticGraph.reln(parent, root);
+                            double weight = newSemanticGraph.getEdge(parent, root).getWeight();
+                            newSemanticGraph.addEdge(parent, newRoot, reln, weight, false);
+                        }
+                    }
+                    //To process verbs if the main conj is the root of the graph. Assign verb as new root of the graph, remove the
oother
+                } else {
+                    isTreeRoot = true;
+
+                    for(IndexedConstituent c : newConstituents) {
+                        c.getSemanticGraph().setRoot(newRoot);
+                        c.setRoot(newRoot);
+                    }
+
+
+                }
+
+
+
+
+                // Checks if the children also belongs to the conjoint and if they do, it assignes
+                // them
+                for(Constituent c: newConstituents) {
+                    SemanticGraph newSemanticGraph = ((IndexedConstituent) c).getSemanticGraph();
+                    List<SemanticGraphEdge> outed = newSemanticGraph.getOutEdgesSorted(root);
+                    for (SemanticGraphEdge ed : outed) {
+                        IndexedWord child = ed.getDependent();
+                        if(DpUtils.isPredet(ed) && ed.getDependent().lemma().equals("both")) { //if it is one level down
+                            for(Constituent cc: constituents) {
+                                if(((IndexedConstituent) cc).getSemanticGraph().containsEdge(ed))
+                                    ((IndexedConstituent) cc).getSemanticGraph().removeEdge(ed);
+                            }
+                        } else if (!DpUtils.isAnyConj(ed) && !DpUtils.isCc(ed) && !DpUtils.isPreconj(ed)
+                                //  && isDescendant(depTree, newRoot.index(), root.index(), child.index())) {
+                                && isDescendant(depTree, newSemanticGraph, newRoot, root, child)) {
+                            GrammaticalRelation reln = newSemanticGraph.reln(root, child);
+                            double weight = newSemanticGraph.getEdge(root, child).getWeight();
+                            newSemanticGraph.addEdge(newRoot, child, reln, weight, false);
+
+                        }
+                    }
+                }
+
+
+
+
+
+
+                // disconect the root of the conjoint from the new graph
+                for(Constituent c: newConstituents) {
+                    SemanticGraph newSemanticGraph = ((IndexedConstituent) c).getSemanticGraph();
+                    List<SemanticGraphEdge> inedges = newSemanticGraph.getIncomingEdgesSorted(root);
+                    for (SemanticGraphEdge redge : inedges)
+                        newSemanticGraph.removeEdge(redge);
+                }
+
+                for(Constituent c: constituents) {
+                    ((IndexedConstituent) c).getSemanticGraph().removeEdge(edge);
+                }
+
+                if(isTreeRoot) {
+                    for(IndexedConstituent c : newConstituents) {
+                        c.getSemanticGraph().removeVertex(root);
+                    }
+                }
```

```
+
+                       constituents.addAll(newConstituents);
+                       //    for(IndexedConstituent c: newConstituents) {
+                       //                  c.getSemanticGraph().prettyPrint();
+                       // }
+
+                       // It passes the constituent with the correct root, if it is the first level it
+                       // should be the new constituent
+
+                       if(levels != null)
+                            levels = levels -1;
+
+                       if (firstLevel || completeProcess) {
+                            generateConstituents(((IndexedConstituent) newConstituents.get(0)).getSemanticGraph(), depTree,
newConstituents.get(0), newRoot,
+                                      constituents, false, processVerb, completeProcess, levels, clauseRoot, roots);
+                       } else {
+                            generateConstituents(((IndexedConstituent) newConstituents.get(0)).getSemanticGraph(), depTree, constituent,
newRoot,
+                                      constituents, false, processVerb, completeProcess, levels, clauseRoot, roots);
+                       }
+
+
+
+                       // deletes the edge containing the conjunction e.g. and, or, but, etc
+                  } else if ((DpUtils.isCc(edge) || DpUtils.isPreconj(edge))&& processCC && !edge.getDependent().lemma().equals("&")) {
+                       for(Constituent c: constituents) {
+                            ((IndexedConstituent) c).getSemanticGraph().removeEdge(edge);
+                       }
+                  } else if(!DpUtils.isPredet(edge) && !constituent.excludedVertexes.contains(edge.getDependent()))
+                       if(levels != null)
+                            levels = levels -1;
+                  generateConstituents(semanticGraph, depTree, constituent, edge.getDependent(),
+                            constituents, false, processVerb, completeProcess, levels, clauseRoot, roots);
+             }
+
+       }
+
+       private static boolean shareAllAncestors(SemanticGraph semanticGraph1, SemanticGraph semanticGraph2, IndexedWord root) {
+
+             Set<IndexedWord> d2 = semanticGraph2.descendants(root);
+             if(d2 == null || d2.isEmpty()) {
+                  return false;
+             }
+
+             Set<IndexedWord> d1 = semanticGraph1.descendants(root);
+
+             Set<IndexedWord> v1 = semanticGraph1.descendants(semanticGraph1.getFirstRoot()); //Assumes only one root, otherwhise
one could delete the non used nodes and call vertexset
+             Set<IndexedWord> v2 = semanticGraph2.descendants(semanticGraph2.getFirstRoot());
+
+             int asize1 = v1.size() - d1.size();
+             int asize2 = v2.size() - d2.size();
+
+             if(asize1 != asize2)
+                  return false;
+
+             for(IndexedWord v: v1) {
+                  if(d1.contains(v))
+                       continue;
+                  if(!v2.contains(v))
+                       return false;
+             }
+             return true;
+       }
+
+       private static boolean containsDescendant(Tree parse, SemanticGraph semanticGraphRoot, SemanticGraph semanticGraphNew,
IndexedWord root, IndexedWord newRoot) {
+             Set<IndexedWord> d1 = semanticGraphRoot.descendants(root);
```

```
+           Set<IndexedWord> d2 = semanticGraphNew.descendants(root);
+
+           for(IndexedWord w: d1) {
+                //if(!d2.contains(w) && isDescendant(parse, newRoot.index(), root.index(), w.index()) )
+                if(isDescendant(parse, semanticGraphRoot, newRoot, root, w))
+                     return true;
+           }
+
+           return false;
+
+      }
+
+
+
+      /** Checks if a node depending on one conjoint also depends to the other */
+      //"He buys and sells electronic products" "Is products depending on both sells and buys?"
+      private static boolean isDescendant(Tree parse, int indexCheck, int indexPivot,
+                                int indexElement) {
+           Tree pivot = parse.getLeaves().get(indexPivot - 1); // because tree parse indexing system
+           // starts with 0
+           Tree check = parse.getLeaves().get(indexCheck - 1);
+           Tree element = parse.getLeaves().get(indexElement - 1);
+
+           while ((!element.value().equals("ROOT"))) {// find a common parent between the head conjoint
+                // and the constituent of the element
+                if (element.pathNodeToNode(element, pivot) != null) // is this efficient enough?
+                     break;
+                element = element.parent(parse);
+           }
+
+           List<Tree> path = element.pathNodeToNode(element, check); // find a path between the common
+           // parent and the other conjoint
+
+           if (path != null)
+                return true;
+           else
+                return false;
+      }
+
+
+      /** Checks if a node depending on one conjoint also depends to the other */
+      //"He buys and sells electronic products" "Is products depending on both sells and buys?"
+      public static boolean isDescendant(Tree parse, SemanticGraph semanticGraph, IndexedWord checkIW, IndexedWord pivotIW,
+                                IndexedWord elementIW) {
+           Tree pivot = DpUtils.getNode(pivotIW, parse, semanticGraph);
+           Tree check = DpUtils.getNode(checkIW, parse, semanticGraph);
+           Tree element = DpUtils.getNode(elementIW, parse, semanticGraph);
+
+           while ((!element.value().equals("ROOT"))) {// find a common parent between the head conjoint
+                // and the constituent of the element
+                if (element.pathNodeToNode(element, pivot) != null) // is this efficient enough?
+                     break;
+                element = element.parent(parse);
+           }
+
+           List<Tree> path = element.pathNodeToNode(element, check); // find a path between the common
+           // parent and the other conjoint
+
+           if (path != null)
+                return true;
+           else
+                return false;
+      }
+
+
+
+      /** Retrieves the heads of the clauses according to the CCs processing options. The result contains
+       * verbs conjoined and a complement if it is conjoined with a verb.*/
+      public static List<IndexedWord> getIndexedWordsConj(SemanticGraph semanticGraph, Tree depTree,
```

```
+                                       IndexedWord root, GrammaticalRelation rel, List<SemanticGraphEdge> toRemove,
+                                       Options option) {
+           List<IndexedWord> ccs = new ArrayList<IndexedWord>(); // to store the conjoints
+           ccs.add(root);
+           List<SemanticGraphEdge> outedges = semanticGraph.outgoingEdgeList(root);
+           for (SemanticGraphEdge edge : outedges) {
+                if (edge.getRelation().equals(rel)) {
+                     List<SemanticGraphEdge> outed = semanticGraph
+                          .outgoingEdgeList(edge.getDependent());
+                     // first condition tests if verbs are involved in the conjoints. Conjunctions between complements are treated elsewhere.
+                     boolean ccVerbs = edge.getDependent().tag().charAt(0) == 'V'
+                          || edge.getGovernor().tag().charAt(0) == 'V';
+                     //This condition will check if there is a cop conjoined with a verb
+                     boolean ccCop = DpUtils.findFirstOfRelationOrDescendent(outed,
+                                                 EnglishGrammaticalRelations.COPULA) != null;
+                     // this condition checks if there are two main clauses conjoined by the CC
+                     boolean ccMainClauses = DpUtils.findFirstOfRelationOrDescendent(outed,
+                                                       EnglishGrammaticalRelations.SUBJECT) != null ||
DpUtils.findFirstOfRelationOrDescendent(outed,
+
EnglishGrammaticalRelations.EXPLETIVE) != null;
+
+                     // This flag will check if the cc should be processed according to the flag and the
+                     // shared elements.
+                     boolean notProcess = !option.processCcAllVerbs && outed.isEmpty()
+                          && shareAll(outedges, depTree, semanticGraph, root, edge.getDependent());
+
+                     if ((ccVerbs || ccCop) && !ccMainClauses && !notProcess) {
+                          ccs.add(edge.getDependent());
+                     }
+
+                     // Disconnects the conjoints. Independent clauses are always disconnected.
+                     if (((ccVerbs || ccCop) && !notProcess) || ccMainClauses) {
+                          toRemove.add(edge);
+
+                          //To remove the coordination
+                          if (option.processCcAllVerbs || !notProcess) {
+                               List<SemanticGraphEdge> conjunct = DpUtils.getEdges(outedges,
+                                                           EnglishGrammaticalRelations.COORDINATION);
+                               for (SemanticGraphEdge e : conjunct) {
+                                    if (e.getDependent().index() > edge.getDependent().index())
+                                         continue;
+                                    if (nextToVerb(depTree, semanticGraph, root, edge.getDependent(), e.getDependent())) {
+//                                    nextToVerb(depTree, root.index(), edge.getDependent().index(), e
+//                                .getDependent().index())) {
+                                         toRemove.add(e);
+                                         break;
+                                    }
+                               }
+                          }
+                     }
+                }
+           }
+           if(ccs.size() > 1)
+                rewriteGraph(semanticGraph, depTree, ccs);
+           return ccs;
+      }
+
+      /** Rewrites the graph so that each conjoint is independent from each other.
+       * They will be disconnected and each dependent correspondignly assigned */
+      private static void rewriteGraph(SemanticGraph semanticGraph, Tree depTree,
+                                 List<IndexedWord> ccs) {
+
+           for(int i = 0; i < ccs.size(); i++) {
+                for(int j = i + 1; j < ccs.size(); j++) {
+                     //Connect each node in ccs to its parent
+                     for (SemanticGraphEdge ed : semanticGraph.getIncomingEdgesSorted(ccs.get(i))) {
+                          if(semanticGraph.getParents(ccs.get(j)).contains(ed.getGovernor())) continue;
+                          semanticGraph.addEdge(ed.getGovernor(), ccs.get(j), ed.getRelation(), ed.getWeight(), false);
```

```
+                     }
+
+                     //Check if the dependents of the main conjoint are also dependent on each of the conjoints
+                     // and assign them in each case.
+                     for (SemanticGraphEdge ed : semanticGraph.getOutEdgesSorted(ccs.get(i))) {
+                          IndexedWord child = ed.getDependent();
+                          if(semanticGraph.getChildren(ccs.get(j)).contains(child)) continue;
+                          if (!DpUtils.isAnyConj(ed) && !DpUtils.isCc(ed)
+                             //&& isDescendant(depTree, ccs.get(j).index(), ccs.get(i).index(), child.index())) {
+                             && isDescendant(depTree, semanticGraph, ccs.get(j), ccs.get(i), child)) {
+                               semanticGraph.addEdge(ccs.get(j), child, ed.getRelation(), ed.getWeight(), false);
+                          }
+                     }
+
+                }
+           }
+      }
+
+
+      private static boolean nextToVerb(Tree depTree, SemanticGraph semanticGraph, IndexedWord firstVerb, IndexedWord
secondVerb, IndexedWord conj) {
+           Tree fverb = DpUtils.getNode(firstVerb, depTree, semanticGraph);
+           Tree sverb = DpUtils.getNode(secondVerb, depTree, semanticGraph);
+           Tree conjv = DpUtils.getNode(conj, depTree, semanticGraph);
+
+           // This will lead us to the level in the tree we want to compare
+           conjv = conjv.parent(depTree);
+
+
+           List<Tree> siblings = conjv.siblings(depTree);
+           Tree[] children = conjv.parent(depTree).children();
+           if (children.length == 0)
+                return false;
+
+           // This will give the node of the conjoint dominating the coordination
+           while (!siblings.contains(fverb)) {
+                fverb = fverb.parent(depTree);
+                if (fverb.equals(depTree))
+                     return false;
+           }
+
+           // same for the other conjoint
+           while (!siblings.contains(sverb)) {
+                sverb = sverb.parent(depTree);
+                if (sverb.equals(depTree))
+                     return false;
+           }
+
+           Integer fv = null;
+           Integer sv = null;
+
+           // This will take the indexes of the nodes dominating the conjoint
+           for (int i = 0; i < children.length; i++) {
+                if (children[i].equals(fverb))
+                     fv = i;
+                else if (children[i].equals(sverb))
+                     sv = i;
+                if (fv != null & sv != null)
+                     break;
+           }
+
+
+           // This will check if they are continuous
+           if(fv == null || sv == null)
+                return false;
+           //Assumes that the minimum distance between adjacent conjoints is 2 in the most usual case---> a,b,c and d
+           //It is <= 3 to work in the case a,b,c,and, d In the last one the distance is 3.
+           else if (sv - fv <= 3)
+                return true;
```

```
+
+           else
+                return false;
+
+      }
+
+      /** Checks if two nodes are conjoined by a given conjunction */
+      private static boolean nextToVerb(Tree depTree, int firstVerb, int secondVerb, int conj) {
+           Tree fverb = depTree.getLeaves().get(firstVerb - 1);
+           Tree sverb = depTree.getLeaves().get(secondVerb - 1);
+           Tree conjv = depTree.getLeaves().get(conj - 1);
+
+           // This will lead us to the level in the tree we want to compare
+           conjv = conjv.parent(depTree);
+
+
+           List<Tree> siblings = conjv.siblings(depTree);
+           Tree[] children = conjv.parent(depTree).children();
+           if (children.length == 0)
+                return false;
+
+           // This will give the node of the conjoint dominating the coordination
+           while (!siblings.contains(fverb)) {
+                fverb = fverb.parent(depTree);
+                if (fverb.equals(depTree))
+                     return false;
+           }
+
+           // same for the other conjoint
+           while (!siblings.contains(sverb)) {
+                sverb = sverb.parent(depTree);
+                if (sverb.equals(depTree))
+                     return false;
+           }
+
+           Integer fv = null;
+           Integer sv = null;
+
+           // This will take the indexes of the nodes dominating the conjoint
+           for (int i = 0; i < children.length; i++) {
+                if (children[i].equals(fverb))
+                     fv = i;
+                else if (children[i].equals(sverb))
+                     sv = i;
+                if (fv != null & sv != null)
+                     break;
+           }
+
+
+           // This will check if they are continuous
+           if(fv == null || sv == null)
+                return false;
+           //Assumes that the minimum distance between adjacent conjoints is 2 in the most usual case---> a,b,c and d
+           //It is <= 3 to work in the case a,b,c,and, d In the last one the distance is 3.
+           else if (sv - fv <= 3)
+                return true;
+
+      else
+         return false;
+
+   }
+
+   /** Checks if two conjoints verbs share all dependents */
+   private static boolean shareAll(List<SemanticGraphEdge> outedges, Tree depTree, SemanticGraph semanticGraph,
+           IndexedWord root, IndexedWord conj) {
+      for (SemanticGraphEdge edge : outedges) {
+         if (DpUtils.isAnySubj(edge) || edge.getDependent().equals(conj))
+            continue;
+        // else if (!isDescendant(depTree, conj.index(), root.index(), edge.getDependent()
```

```
+         //       .index()))
+          else if(isDescendant(depTree, semanticGraph, conj, root, edge.getDependent()))
+             return false;
+      }
+
+      return true;
+   }
+
+}
diff --git a/src/clausie/Proposition.java b/src/clausie/Proposition.java
new file mode 100644
index 0000000..3aa2653
--- /dev/null
+++ b/src/clausie/Proposition.java
@@ -0,0 +1,127 @@
+package clausie;
+
+import java.util.ArrayList;
+import java.util.HashSet;
+import java.util.List;
+import java.util.Set;
+
+/**
+ * Stores a proposition.
+ *
+ * @date $LastChangedDate: 2014-05-20 15:17:22 +0200 (Tue, 20 May 2014) $
+ * @version $LastChangedRevision: 1601 $
+ */
+public class Proposition {
+
+	/** Constituents of the proposition */
+	public List<String> constituents = new ArrayList<String>();
+
+	/** Position of optional constituents */
+	Set<Integer> optional = new HashSet<Integer>();
+
+	// TODO: types of constituents (e.g., optionality)
+	// sentence ID etc.
+
+	public Proposition() {
+	}
+
+	/** Returns the subject of the proposition */
+	public String subject() {
+		return constituents.get(0);
+	}
+
+	/** Returns the relation of the proposition */
+	public String relation() {
+		return constituents.get(1);
+	}
+
+	/**
+	 * Returns a constituent in a given position <br>
+	 * SVO => i=0 is O
+	 */
+	public String argument(int i) {
+		return constituents.get(i + 2);
+	}
+
+	/** Returns the number of arguments */
+	public int noArguments() {
+		return constituents.size() - 2;
+	}
+
+	/** Checks if an argument is optional */
+	public boolean isOptionalArgument(int i) {
+		return optional.contains(i + 2);
+	}
```

```
+
+	/**
+	 * SVO => i=0 is S, i=1 is V, ...
+	 * @return
+	 */
+	public List<String> getConstituents() {
+		return constituents;
+	}
+
+	public void setConstituents(List<String> constituents) {
+		this.constituents = constituents;
+	}
+
+	@Override
+	public String toString() {
+		StringBuffer sb = new StringBuffer();
+		String sep = "(";
+		for (int i = 0; i < constituents.size(); i++) {
+			String constituent = constituents.get(i);
+			sb.append(sep);
+			sep = ", ";
+			sb.append("\"");
+			sb.append(constituent);
+			sb.append("\"");
+			if (optional.contains(i)) {
+				sb.append("?");
+			}
+		}
+		sb.append(")");
+		return sb.toString();
+	}
+
+	@Override
+	public int hashCode() {
+		final int prime = 31;
+		int result = 1;
+		result = prime * result
+				+ ((constituents == null) ? 0 : constituents.hashCode());
+		result = prime * result
+				+ ((optional == null) ? 0 : optional.hashCode());
+		return result;
+	}
+
+	@Override
+	public boolean equals(Object obj) {
+		if (this == obj)
+			return true;
+		if (obj == null)
+			return false;
+		if (getClass() != obj.getClass())
+			return false;
+		Proposition other = (Proposition) obj;
+		if (constituents == null) {
+			if (other.constituents != null)
+				return false;
+		} else if (!constituents.equals(other.constituents))
+			return false;
+		if (optional == null) {
+			if (other.optional != null)
+				return false;
+		} else if (!optional.equals(other.optional))
+			return false;
+		return true;
+	}
+
+	@Override
+	public Proposition clone() {
+		Proposition clone = new Proposition();
+		clone.constituents = new ArrayList<String>(this.constituents);
```

```
+                           clone.optional = new HashSet<Integer>(this.optional);
+                           return clone;
+               }
+}
diff --git a/src/clausie/PropositionGenerator.java b/src/clausie/PropositionGenerator.java
new file mode 100644
index 0000000..f220fc8
--- /dev/null
+++ b/src/clausie/PropositionGenerator.java
@@ -0,0 +1,108 @@
+package clausie;
+
+import java.util.Collection;
+import java.util.Collections;
+import java.util.HashSet;
+import java.util.List;
+import java.util.Set;
+import java.util.TreeSet;
+
+import edu.stanford.nlp.ling.IndexedWord;
+import edu.stanford.nlp.semgraph.SemanticGraph;
+import edu.stanford.nlp.trees.EnglishGrammaticalRelations;
+import edu.stanford.nlp.trees.GrammaticalRelation;
+
+/** Handles the generation of propositions out of a given clause
+ *
+ * @date $ $
+ * @version $ $ */
+public abstract class PropositionGenerator {
+
+       Options options;
+
+       /** Relations to be excluded in every constituent of a clause except the verb */
+       protected static final Set<GrammaticalRelation> EXCLUDE_RELATIONS;
+
+       /** Relations to be excluded in the verb */
+       protected static final Set<GrammaticalRelation> EXCLUDE_RELATIONS_VERB;
+
+       static {
+               EXCLUDE_RELATIONS = new HashSet<GrammaticalRelation>();
+               EXCLUDE_RELATIONS.add(EnglishGrammaticalRelations.RELATIVE_CLAUSE_MODIFIER);
+               EXCLUDE_RELATIONS.add(EnglishGrammaticalRelations.APPOSITIONAL_MODIFIER);
+               EXCLUDE_RELATIONS.add(EnglishGrammaticalRelations.PARATAXIS);
+
+               EXCLUDE_RELATIONS_VERB = new HashSet<GrammaticalRelation>();
+               EXCLUDE_RELATIONS_VERB.addAll(EXCLUDE_RELATIONS);
+               EXCLUDE_RELATIONS_VERB.add(EnglishGrammaticalRelations.valueOf("dep")); //without this asome adverbs or auxiliaries
will end up in the relation
+       }
+
+       /** Constructs a proposition generator*/
+       public PropositionGenerator(Options options) {
+               this.options = options;
+       }
+
+       /** Generates propositions for a given clause*/
+       public abstract void generate(List<Proposition> result, Clause clause, List<Boolean> include);
+
+       /** Generates a textual representation of a given constituent plus a set of words*/
+       private String generatePhrase(IndexedConstituent constituent, Collection<IndexedWord> words) {
+               StringBuffer result = new StringBuffer();
+               String separator = "";
+               result.append(separator);
+               if (constituent.isPrepositionalPhrase()) {
+                       if (options.lemmatize) {
+                               result.append(constituent.getRoot().lemma());
+                       } else {
+                               result.append(constituent.getRoot().word());
+                       }
```

```
+                separator = " ";
+            }
+
+            for (IndexedWord word : words) {
+                result.append(separator);
+                if (options.lemmatize) {
+                    result.append(word.lemma());
+                } else {
+                    result.append(word.word());
+                }
+                separator = " ";
+            }
+            return result.toString();
+        }
+
+        /** Generates a textual representation of a given constituent in a given clause*/
+        public String generate(Clause clause, int constituentIndex) {
+            Set<GrammaticalRelation> excludeRelations = EXCLUDE_RELATIONS;
+            if (clause.getVerb() == constituentIndex) {
+                excludeRelations = EXCLUDE_RELATIONS_VERB;
+            }
+            return generate(clause, constituentIndex, excludeRelations,
+                    Collections.<GrammaticalRelation> emptySet());
+        }
+
+        /** Generates a textual representation of a given constituent in a given clause*/
+        public String generate(Clause clause, int constituentIndex,
+                    Collection<GrammaticalRelation> excludeRelations,
+                    Collection<GrammaticalRelation> excludeRelationsTop) {
+            Constituent constituent = clause.getConstituents().get(constituentIndex);
+            if (constituent instanceof TextConstituent) {
+                return ((TextConstituent) constituent).text();
+            } else if (constituent instanceof IndexedConstituent) {
+                IndexedConstituent iconstituent = (IndexedConstituent) constituent;
+                SemanticGraph subgraph = iconstituent.createReducedSemanticGraph();
+                DpUtils.removeEdges(subgraph, iconstituent.getRoot(),
+              excludeRelations, excludeRelationsTop);
+          Set<IndexedWord> words = new TreeSet<IndexedWord>(
+                subgraph.descendants(iconstituent.getRoot()));
+          for (IndexedWord v : iconstituent.getAdditionalVertexes()) {
+            words.addAll(subgraph.descendants(v));
+          }
+          if (iconstituent.isPrepositionalPhrase())
+            words.remove(iconstituent.getRoot());
+          return generatePhrase(iconstituent, words);
+        } else {
+          throw new IllegalArgumentException();
+        }
+    }
+}
diff --git a/src/clausie/RelativeClauseIndexedConstituent.java b/src/clausie/RelativeClauseIndexedConstituent.java
new file mode 100644
index 0000000..c24e59a
--- /dev/null
+++ b/src/clausie/RelativeClauseIndexedConstituent.java
@@ -0,0 +1,49 @@
+package clausie;
+
+import java.util.List;
+import java.util.Set;
+
+import edu.stanford.nlp.ling.IndexedWord;
+import edu.stanford.nlp.semgraph.SemanticGraph;
+
+public class RelativeClauseIndexedConstituent extends IndexedConstituent {
+
+private List<Clause> clauses;
+
+
```

```
+	private RelativeClauseIndexedConstituent() {
+
+	}
+
+
+	public RelativeClauseIndexedConstituent(SemanticGraph semanticGraph, IndexedWord root,
+				Type type, List<Clause> clauses) {
+		super(semanticGraph, root, type);
+		this.setClauses(clauses);
+	}
+
+
+	public RelativeClauseIndexedConstituent(SemanticGraph semanticGraph, IndexedWord root,
+				Set<IndexedWord> additionalVertexes,
+				Set<IndexedWord> excludedVertexes, Type type, List<Clause> clauses) {
+		super(semanticGraph, root, additionalVertexes, excludedVertexes, type);
+		this.setClauses(clauses);
+	}
+
+	/** Returns the clauses derived from the constituent. */
+	public List<Clause> getClauses() {
+		return clauses;
+	}
+
+	/** Sets the clauses derived from the constituent. */
+	public void setClauses(List<Clause> clauses) {
+		this.clauses = clauses;
+	}
+
+
+	 public static void main(String[] args) {
+			// TODO Auto-generated method stub
+
+	 }
+
+}
diff --git a/src/clausie/StructuredConstituent.java b/src/clausie/StructuredConstituent.java
new file mode 100644
index 0000000..3e0c452
--- /dev/null
+++ b/src/clausie/StructuredConstituent.java
@@ -0,0 +1,187 @@
+package clausie;
+
+import java.util.ArrayList;
+import java.util.List;
+import java.util.Set;
+
+import edu.stanford.nlp.ling.IndexedWord;
+import edu.stanford.nlp.semgraph.SemanticGraph;
+import edu.stanford.nlp.semgraph.SemanticGraphEdge;
+
+public class StructuredConstituent extends IndexedConstituent {
+
+	List<Constituent> constituents = new ArrayList<Constituent>();
+
+	private Integer mark = null;
+
+	private Integer aux = null;
+
+	List<Integer> senses = new ArrayList<Integer>();
+
+	List<Integer> prepPhrases = new ArrayList<Integer>();
+
+	List<Integer> xcomp = new ArrayList<Integer>();
+
+	List<Integer> advcl = new ArrayList<Integer>();
+
+	List<Integer> advmods = new ArrayList<Integer>();
```

```
+
+	private List<Integer> amods = new ArrayList<Integer>();
+
+	private List<Integer> nn = new ArrayList<Integer>();
+
+	List<Integer> appos = new ArrayList<Integer>();
+
+	List<Integer> conj = new ArrayList<Integer>();
+
+	List<Integer> infmod = new ArrayList<Integer>();
+
+	List<Integer> partmod = new ArrayList<Integer>();
+
+	List<Integer> rcmod = new ArrayList<Integer>();
+
+	private List<Integer> subjects = new ArrayList<Integer>();
+
+	private List<Integer> ccomp = new ArrayList<Integer>();
+
+
+	public StructuredConstituent(SemanticGraph semanticGraph, IndexedWord root,
+				Type type) {
+		super(semanticGraph, root, type);
+	}
+	/** Constructs a new indexed constituent for any noun-phrase relation.
+	 *
+	 * @param semanticGraph Semantic graph for this constituent ({@see #semanticGraph})
+	 * @param root The root vertex of this constituent ({@see {@link #root})
+	 * @param additionalVertexes Additional root vertexes that form this constituent ({@see
+	 *		{@link #additionalVertexes})
+	 * @param excludedVertexes Vertexes that are excluded from this constituent ({@see
+	 *		{@link #excludedVertexes})
+	 * @param type type of this constituent
+	 * * @param clauses derived from this constituent*/
+	public StructuredConstituent(SemanticGraph semanticGraph, IndexedWord root,
+				Set<IndexedWord> additionalVertexes,
+				Set<IndexedWord> excludedVertexes, Type type) {
+		super(semanticGraph, root, additionalVertexes, excludedVertexes, type);
+	}
+
+
+
+	public StructuredConstituent build() {
+		List<SemanticGraphEdge> edges = semanticGraph.outgoingEdgeList(root);
+		for(SemanticGraphEdge edge: edges) {
+			if(DpUtils.isAnyPrep(edge)) {
+				getConstituents().add(new PrepositionalPhraseIndexedConstituent(semanticGraph, edge.getDependent(),
null).build());
+				getPrepPhrases().add(getConstituents().size() - 1);
+			} else if(DpUtils.isAdvmod(edge)) {
+				getConstituents().add(new IndexedConstituent(semanticGraph, edge.getDependent(), null));
+				advmods.add(getConstituents().size() - 1);
+			} else if(DpUtils.isAMod(edge)) {
+				getConstituents().add(new IndexedConstituent(semanticGraph, edge.getDependent(), null));
+				getAmods().add(getConstituents().size() - 1);
+			} else if(DpUtils.isNN(edge)) {
+				getConstituents().add(new IndexedConstituent(semanticGraph, edge.getDependent(), null));
+				getNn().add(getConstituents().size() - 1);
+			} else if(DpUtils.isAppos(edge)) {
+				getConstituents().add(new IndexedConstituent(semanticGraph, edge.getDependent(), null));
+				appos.add(getConstituents().size() - 1);
+			} else if(DpUtils.isAnyConj(edge)) {
+				getConstituents().add(new IndexedConstituent(semanticGraph, edge.getDependent(), null));
+				conj.add(getConstituents().size() - 1);
+				//} else if(DpUtils.isInfmod(edge)) {
+				//  getConstituents().add(new IndexedConstituent(semanticGraph, edge.getDependent(), null));
+				//  infmod.add(getConstituents().size() - 1);
+				//} else if(DpUtils.isPartMod(edge)) {
+				//  getConstituents().add(new IndexedConstituent(semanticGraph, edge.getDependent(), null));
```

```
+                    //  partmod.add(getConstituents().size() - 1);
+                } else if(DpUtils.isRcmod(edge)) {
+                    getConstituents().add(new IndexedConstituent(semanticGraph, edge.getDependent(), null));
+                    rcmod.add(getConstituents().size() - 1);
+                } else if(DpUtils.isAdvcl(edge)) {
+                    //anothe clause check
+                    getConstituents().add(new IndexedConstituent(semanticGraph, edge.getDependent(), null));
+                    advcl.add(getConstituents().size() - 1);
+                } else if(DpUtils.isMark(edge)) {
+                    //anothe clause check
+                    getConstituents().add(new IndexedConstituent(semanticGraph, edge.getDependent(), null));
+                    setMark(getConstituents().size() - 1);
+                } else if(DpUtils.isAux(edge)) {
+                    //anothe clause check
+                    getConstituents().add(new IndexedConstituent(semanticGraph, edge.getDependent(), null));
+                    setAux(getConstituents().size() - 1);
+                } else if(DpUtils.isAnySubj(edge)) {
+                    //anothe clause check
+                    getConstituents().add(new IndexedConstituent(semanticGraph, edge.getDependent(), null));
+                    getSubjects().add(getConstituents().size() - 1);
+                } else if(DpUtils.isCcomp(edge)) {
+                    //anothe clause check
+                    getConstituents().add(new IndexedConstituent(semanticGraph, edge.getDependent(), null));
+                    getCcomp().add(getConstituents().size() - 1);
+                } else if(DpUtils.isXcomp(edge)) {
+                    //anothe clause check
+                    getConstituents().add(new StructuredConstituent(semanticGraph, edge.getDependent(), null).build());
+                    getXcomps().add(getConstituents().size() - 1);
+                }
+            }
+            return this;
+        }
+
+         public List<Integer> getPrepPhrases() {
+                    return prepPhrases;
+         }
+
+
+         public void setPrepPhrases(List<Integer> prepPhrases) {
+                    this.prepPhrases = prepPhrases;
+         }
+
+
+         public List<Constituent> getConstituents() {
+                    return constituents;
+         }
+
+
+         public void setConstituents(List<Constituent> constituents) {
+                    this.constituents = constituents;
+         }
+         public List<Integer> getNn() {
+                    return nn;
+         }
+         public void setNn(List<Integer> nn) {
+                    this.nn = nn;
+         }
+         public List<Integer> getAmods() {
+                    return amods;
+         }
+         public void setAmods(List<Integer> amods) {
+                    this.amods = amods;
+         }
+         public Integer getMark() {
+                    return mark;
+         }
+         public void setMark(Integer mark) {
+                    this.mark = mark;
+         }
```

```
+		public Integer getAux() {
+			return aux;
+		}
+		public void setAux(Integer aux) {
+			this.aux = aux;
+		}
+		public List<Integer> getSubjects() {
+			return subjects;
+		}
+		public void setSubjects(List<Integer> subjects) {
+			this.subjects = subjects;
+		}
+		public List<Integer> getCcomp() {
+			return ccomp;
+		}
+		public List<Integer> getXcomps() {
+			return xcomp;
+		}
+		public void setCcomp(List<Integer> ccomp) {
+			this.ccomp = ccomp;
+		}
+
+}
diff --git a/src/clausie/TextConstituent.java b/src/clausie/TextConstituent.java
new file mode 100644
index 0000000..a2fcb36
--- /dev/null
+++ b/src/clausie/TextConstituent.java
@@ -0,0 +1,28 @@
+package clausie;
+
+
+/** A textual expression of a constituent. The constituent is represented as a plain string
+ *  without identifying its internal structure
+ *
+ * @date $ $
+ * @version $ $ */
+public class TextConstituent extends Constituent {
+	String text;
+
+		/** Constructs a constituent with a specified textual representation and type. */
+		public TextConstituent(String text, Type type) {
+			super(type);
+			this.text = text;
+		}
+
+		/** Returns a textual representation of the constituent. */
+		public String text() {
+			return text;
+		}
+
+		/** Returns a textual representation of the constituent. */
+		public String rootString() {
+			return text;
+		}
+
+}
diff --git a/src/clausie/XcompConstituent.java b/src/clausie/XcompConstituent.java
new file mode 100644
index 0000000..6e4dceb
--- /dev/null
+++ b/src/clausie/XcompConstituent.java
@@ -0,0 +1,71 @@
+package clausie;
+
+import java.util.ArrayList;
+import java.util.List;
+import java.util.Set;
+import java.util.TreeSet;
```

```
+
+import edu.stanford.nlp.ling.IndexedWord;
+import edu.stanford.nlp.semgraph.SemanticGraph;
+
+
+/** An {@code XcompConstituent} of a clause formed out of an xcomp.
+ *
+ * Note that the xcomp relation refers to a clause with an external subject.
+ * The constituent stores the set of clauses that can be derived from the xcomp
+ * clause.
+ *
+ * @date $LastChangedDate: 2013-04-23 00:04:28 +0200 (Tue, 23 Apr 2013) $
+ * @version $LastChangedRevision: 734 $ */
+public class XcompConstituent extends StructuredConstituent {
+
+	/** Clauses derived from this constituent */
+	private List<Clause> clauses;
+
+	/** Constructs a new constituent for the xcomp relation.
+	 *
+	 * @param semanticGraph Semantic graph for this constituent ({@see #semanticGraph})
+	 * @param root The root vertex of this constituent ({@see {@link #root})
+	 * @param type type of this constituent
+	 * @param clauses derived from this constituent*/
+	public XcompConstituent(SemanticGraph semanticGraph, IndexedWord root,
+			Type type, List<Clause> clauses) {
+		super(semanticGraph, root, type);
+		this.setClauses(clauses);
+	}
+	/** Constructs a new indexed constituent for the xcomp relation.
+	 *
+	 * @param semanticGraph Semantic graph for this constituent ({@see #semanticGraph})
+	 * @param root The root vertex of this constituent ({@see {@link #root})
+	 * @param additionalVertexes Additional root vertexes that form this constituent ({@see
+	 *		{@link #additionalVertexes})
+	 * @param excludedVertexes Vertexes that are excluded from this constituent ({@see
+	 *		{@link #excludedVertexes})
+	 * @param type type of this constituent
+	 * * @param clauses derived from this constituent*/
+	public XcompConstituent(SemanticGraph semanticGraph, IndexedWord root,
+			Set<IndexedWord> additionalVertexes,
+			Set<IndexedWord> excludedVertexes, Type type, List<Clause> clauses) {
+		super(semanticGraph, root, additionalVertexes, excludedVertexes, type);
+		this.setClauses(clauses);
+	}
+
+	/** Returns the clauses derived from the constituent. */
+	public List<Clause> getClauses() {
+		return clauses;
+	}
+
+	/** Sets the clauses derived from the constituent. */
+	public void setClauses(List<Clause> clauses) {
+		this.clauses = clauses;
+	}
+
+	@Override
+	public XcompConstituent clone() {
+		XcompConstituent clone = new XcompConstituent(semanticGraph, root, type, clauses);
+		return clone;
+	}
+
+
+
+}
diff --git a/src/edu/stanford/nlp/ling/AnnotationLookup.java b/src/edu/stanford/nlp/ling/AnnotationLookup.java
index 2cb96ed..1cde937 100644
--- a/src/edu/stanford/nlp/ling/AnnotationLookup.java
+++ b/src/edu/stanford/nlp/ling/AnnotationLookup.java
```

```
@@ -71,6 +71,10 @@ public class AnnotationLookup {
    SECTIONID_KEY(CoreAnnotations.SectionIDAnnotation.class,"sectionID"),
    SECTIONDATE_KEY(CoreAnnotations.SectionDateAnnotation.class,"sectionDate"),

+    DF_TYPE_KEY(CoreAnnotations.DFTypeAnnotation.class, "df-type"),
+    DF_TYPE_ID_KEY(CoreAnnotations.DFTypeIDAnnotation.class, "df-type-id"),
+    MENTION_KEY(CoreAnnotations.MentionAnnotation.class, "mention"),
+
    // Thang Sep13: for Genia NER
    HEAD_KEY(CoreAnnotations.HeadWordStringAnnotation.class, "head"),
    GOVERNOR_KEY(CoreAnnotations.GovernorAnnotation.class, "governor"),
diff --git a/src/edu/stanford/nlp/ling/CoreAnnotations.java b/src/edu/stanford/nlp/ling/CoreAnnotations.java
index f43e816..1a3edc1 100644
--- a/src/edu/stanford/nlp/ling/CoreAnnotations.java
+++ b/src/edu/stanford/nlp/ling/CoreAnnotations.java
@@ -1666,4 +1666,24 @@ public class CoreAnnotations {
  public static class LabelIDAnnotation implements CoreAnnotation<Integer>{
    public Class<Integer> getType() { return Integer.class; }
  }
+
+
+  public static class DFTypeAnnotation implements CoreAnnotation<String> {
+    public Class<String> getType() {
+      return String.class;
+    }
+  }
+
+  public static class DFTypeIDAnnotation implements CoreAnnotation<String> {
+    public Class<String> getType() {
+      return String.class;
+    }
+  }
+
+  public static class MentionAnnotation implements CoreAnnotation<String> {
+    public Class<String> getType() {
+      return String.class;
+    }
+  }
+
 }
diff --git a/src/edu/stanford/nlp/pipeline/Annotator.java b/src/edu/stanford/nlp/pipeline/Annotator.java
index 58cb952..5281eda 100644
--- a/src/edu/stanford/nlp/pipeline/Annotator.java
+++ b/src/edu/stanford/nlp/pipeline/Annotator.java
@@ -6,17 +6,17 @@ import java.util.Collections;
 import java.util.Set;

 /**
- * This is an interface for adding annotations to a partially annotated
- * Annotation.  In some ways, it is just a glorified function, except
- * that it explicitly operates in-place on Annotation objects.  Annotators
+ * This is an interface for adding annotations to a fully annotated
+ * Annotation.  In some ways, it is just a glorified Function, except
+ * that it explicitly operates on Annotation objects.  Annotators
  * should be given to an AnnotationPipeline in order to make
  * annotation pipelines (the whole motivation of this package), and
  * therefore implementers of this interface should be designed to play
  * well with other Annotators and in their javadocs they should
  * explicitly state what annotations they are assuming already exist
- * in the annotation (like parse, POS tag, etc), what keys they are
- * expecting them under (see, for instance, the ones in CoreAnnotations),
- * and what annotations they will add (or modify) and the keys
+ * in the annotation (like parse, POS tag, etc), what field they are
+ * expecting them under (Annotation.WORDS_KEY, Annotation.PARSE_KEY,
+ * etc) and what annotations they will add (or modify) and the keys
  * for them as well.  If you would like to look at the code for a
  * relatively simple Annotator, I recommend NERAnnotator.  For a lot
  * of code you could just add the implements directly, but I recommend
@@ -40,7 +40,7 @@ public interface Annotator {
```

```
   /**
    * Given an Annotation, perform a task on this Annotation.
    */
-  void annotate(Annotation annotation);
+  public void annotate(Annotation annotation) ;

   /**
    * The Requirement is a general way of describing the pre and post
@@ -55,14 +55,14 @@ public interface Annotator {
    * <br>
    * We do nothing to override the equals or hashCode methods.  This
    * means that two Requirements are equal iff they are the same
-   * object.  We do not want to use {@code name} to decide
+   * object.  We do not want to use <code>name</code> to decide
    * equality because a subclass that uses more information, such as
    * the particular kind of tsurgeon used in a hypothetical
    * TsurgeonAnnotator, cannot use a stricter equals() than the
    * superclass.  It is hard to get stricter than ==.
    */
-  class Requirement {
-    private final String name;
+  public class Requirement {
+    public final String name;
     public Requirement(String name) {
       this.name = name;
     }
@@ -76,57 +76,57 @@ public interface Annotator {
    * Returns a set of requirements for which tasks this annotator can
    * provide.  For example, the POS annotator will return "pos".
    */
-  Set<Requirement> requirementsSatisfied();
+  public Set<Requirement> requirementsSatisfied();

   /**
    * Returns the set of tasks which this annotator requires in order
    * to perform.  For example, the POS annotator will return
    * "tokenize", "ssplit".
    */
-  Set<Requirement> requires();
+  public Set<Requirement> requires();

   /**
    * These are annotators which StanfordCoreNLP knows how to create.
    * Add new annotators and/or annotators from other groups here!
    */
-  String STANFORD_TOKENIZE = "tokenize";
-  String STANFORD_CLEAN_XML = "cleanxml";
-  String STANFORD_SSPLIT = "ssplit";
-  String STANFORD_POS = "pos";
-  String STANFORD_LEMMA = "lemma";
-  String STANFORD_NER = "ner";
-  String STANFORD_REGEXNER = "regexner";
-  String STANFORD_ENTITY_MENTIONS = "entitymentions";
-  String STANFORD_GENDER = "gender";
-  String STANFORD_TRUECASE = "truecase";
-  String STANFORD_PARSE = "parse";
-  String STANFORD_DETERMINISTIC_COREF = "dcoref";
-  String STANFORD_COREF = "hcoref";
-  String STANFORD_RELATION = "relation";
-  String STANFORD_SENTIMENT = "sentiment";
-  String STANFORD_COLUMN_DATA_CLASSIFIER = "cdc";
-  String STANFORD_DEPENDENCIES = "depparse";
-  String STANFORD_NATLOG = "natlog";
-  String STANFORD_OPENIE = "openie";
-  String STANFORD_QUOTE = "quote";
+  public static final String STANFORD_TOKENIZE = "tokenize";
+  public static final String STANFORD_CLEAN_XML = "cleanxml";
+  public static final String STANFORD_SSPLIT = "ssplit";
+  public static final String STANFORD_POS = "pos";
```

```
+  public static final String STANFORD_LEMMA = "lemma";
+  public static final String STANFORD_NER = "ner";
+  public static final String STANFORD_REGEXNER = "regexner";
+  public static final String STANFORD_ENTITY_MENTIONS = "entitymentions";
+  public static final String STANFORD_GENDER = "gender";
+  public static final String STANFORD_TRUECASE = "truecase";
+  public static final String STANFORD_PARSE = "parse";
+  public static final String STANFORD_DETERMINISTIC_COREF = "dcoref";
+  public static final String STANFORD_COREF = "hcoref";
+  public static final String STANFORD_RELATION = "relation";
+  public static final String STANFORD_SENTIMENT = "sentiment";
+  public static final String STANFORD_COLUMN_DATA_CLASSIFIER = "cdc";
+  public static final String STANFORD_DEPENDENCIES = "depparse";
+  public static final String STANFORD_NATLOG = "natlog";
+  public static final String STANFORD_OPENIE = "openie";
+  public static final String STANFORD_QUOTE = "quote";


-  Requirement TOKENIZE_REQUIREMENT = new Requirement(STANFORD_TOKENIZE);
-  Requirement CLEAN_XML_REQUIREMENT = new Requirement(STANFORD_CLEAN_XML);
-  Requirement SSPLIT_REQUIREMENT = new Requirement(STANFORD_SSPLIT);
-  Requirement POS_REQUIREMENT = new Requirement(STANFORD_POS);
-  Requirement LEMMA_REQUIREMENT = new Requirement(STANFORD_LEMMA);
-  Requirement NER_REQUIREMENT = new Requirement(STANFORD_NER);
-  Requirement GENDER_REQUIREMENT = new Requirement(STANFORD_GENDER);
-  Requirement TRUECASE_REQUIREMENT = new Requirement(STANFORD_TRUECASE);
-  Requirement PARSE_REQUIREMENT = new Requirement(STANFORD_PARSE);
-  Requirement DEPENDENCY_REQUIREMENT = new Requirement(STANFORD_DEPENDENCIES);
-  Requirement DETERMINISTIC_COREF_REQUIREMENT = new Requirement(STANFORD_DETERMINISTIC_COREF);
-  Requirement COREF_REQUIREMENT = new Requirement(STANFORD_COREF);
-  Requirement RELATION_EXTRACTOR_REQUIREMENT = new Requirement(STANFORD_RELATION);
-  Requirement NATLOG_REQUIREMENT = new Requirement(STANFORD_NATLOG);
-  Requirement OPENIE_REQUIREMENT = new Requirement(STANFORD_OPENIE);
-  Requirement QUOTE_REQUIREMENT = new Requirement(STANFORD_QUOTE);
+  public static final Requirement TOKENIZE_REQUIREMENT = new Requirement(STANFORD_TOKENIZE);
+  public static final Requirement CLEAN_XML_REQUIREMENT = new Requirement(STANFORD_CLEAN_XML);
+  public static final Requirement SSPLIT_REQUIREMENT = new Requirement(STANFORD_SSPLIT);
+  public static final Requirement POS_REQUIREMENT = new Requirement(STANFORD_POS);
+  public static final Requirement LEMMA_REQUIREMENT = new Requirement(STANFORD_LEMMA);
+  public static final Requirement NER_REQUIREMENT = new Requirement(STANFORD_NER);
+  public static final Requirement GENDER_REQUIREMENT = new Requirement(STANFORD_GENDER);
+  public static final Requirement TRUECASE_REQUIREMENT = new Requirement(STANFORD_TRUECASE);
+  public static final Requirement PARSE_REQUIREMENT = new Requirement(STANFORD_PARSE);
+  public static final Requirement DEPENDENCY_REQUIREMENT = new Requirement(STANFORD_DEPENDENCIES);
+  public static final Requirement DETERMINISTIC_COREF_REQUIREMENT = new Requirement(STANFORD_DETERMINISTIC_COREF);
+  public static final Requirement COREF_REQUIREMENT = new Requirement(STANFORD_COREF);
+  public static final Requirement RELATION_EXTRACTOR_REQUIREMENT = new Requirement(STANFORD_RELATION);
+  public static final Requirement NATLOG_REQUIREMENT = new Requirement(STANFORD_NATLOG);
+  public static final Requirement OPENIE_REQUIREMENT = new Requirement(STANFORD_OPENIE);
+  public static final Requirement QUOTE_REQUIREMENT = new Requirement(STANFORD_QUOTE);

   /**
    * These are annotators which StanfordCoreNLP does not know how to
@@ -135,32 +135,30 @@ public interface Annotator {
    * already included in other parts of the system, such as sutime,
    * which is already included in ner.
    */
-  Requirement GUTIME_REQUIREMENT = new Requirement("gutime");
-  Requirement SUTIME_REQUIREMENT = new Requirement("sutime");
-  Requirement HEIDELTIME_REQUIREMENT = new Requirement("heideltime");
-  Requirement STEM_REQUIREMENT = new Requirement("stem");
-  Requirement NUMBER_REQUIREMENT = new Requirement("number");
-  Requirement TIME_WORDS_REQUIREMENT = new Requirement("timewords");
-  Requirement QUANTIFIABLE_ENTITY_NORMALIZATION_REQUIREMENT = new Requirement("quantifiable_entity_normalization");
-  Requirement COLUMN_DATA_CLASSIFIER = new Requirement("column_data_classifer");
+  public static final Requirement GUTIME_REQUIREMENT = new Requirement("gutime");
+  public static final Requirement SUTIME_REQUIREMENT = new Requirement("sutime");
+  public static final Requirement HEIDELTIME_REQUIREMENT = new Requirement("heideltime");
+  public static final Requirement STEM_REQUIREMENT = new Requirement("stem");
```

```
+  public static final Requirement NUMBER_REQUIREMENT = new Requirement("number");
+  public static final Requirement TIME_WORDS_REQUIREMENT = new Requirement("timewords");
+  public static final Requirement QUANTIFIABLE_ENTITY_NORMALIZATION_REQUIREMENT = new
Requirement("quantifiable_entity_normalization");
+  public static final Requirement COLUMN_DATA_CLASSIFIER = new Requirement("column_data_classifer");

   /**
-   * The Stanford Parser can produce this if it is specifically requested.
+   * The Stanford Parser can produce this if it is specifically requested
    */
-  Requirement BINARIZED_TREES_REQUIREMENT = new Requirement("binarized_trees");
+  public static final Requirement BINARIZED_TREES_REQUIREMENT = new Requirement("binarized_trees");

   /**
    * These are typical combinations of annotators which may be used as
    * requirements by other annotators.
    */
-  Set<Requirement> TOKENIZE_AND_SSPLIT = Collections.unmodifiableSet(new ArraySet<Requirement>(TOKENIZE_REQUIREMENT,
SSPLIT_REQUIREMENT));
-  Set<Requirement> TOKENIZE_SSPLIT_POS = Collections.unmodifiableSet(new ArraySet<Requirement>(TOKENIZE_REQUIREMENT,
SSPLIT_REQUIREMENT, POS_REQUIREMENT));
-  Set<Requirement> TOKENIZE_SSPLIT_NER = Collections.unmodifiableSet(new ArraySet<Requirement>(TOKENIZE_REQUIREMENT,
SSPLIT_REQUIREMENT, NER_REQUIREMENT));
-  Set<Requirement> TOKENIZE_SSPLIT_PARSE = Collections.unmodifiableSet(new ArraySet<Requirement>(TOKENIZE_REQUIREMENT,
SSPLIT_REQUIREMENT, PARSE_REQUIREMENT));
-  Set<Requirement> TOKENIZE_SSPLIT_PARSE_NER = Collections.unmodifiableSet(new
ArraySet<Requirement>(TOKENIZE_REQUIREMENT, SSPLIT_REQUIREMENT, PARSE_REQUIREMENT, NER_REQUIREMENT));
-  Set<Requirement> TOKENIZE_SSPLIT_POS_LEMMA = Collections.unmodifiableSet(new
ArraySet<Requirement>(TOKENIZE_REQUIREMENT, SSPLIT_REQUIREMENT, POS_REQUIREMENT, LEMMA_REQUIREMENT));
-  Set<Requirement> TOKENIZE_SSPLIT_POS_DEPPARSE = Collections.unmodifiableSet(new
ArraySet<Requirement>(TOKENIZE_REQUIREMENT, SSPLIT_REQUIREMENT, POS_REQUIREMENT, DEPENDENCY_REQUIREMENT));
-  Set<Requirement> PARSE_AND_TAG = Collections.unmodifiableSet(new ArraySet<Requirement>(POS_REQUIREMENT,
PARSE_REQUIREMENT));
-  Set<Requirement> PARSE_TAG_BINARIZED_TREES = Collections.unmodifiableSet(new ArraySet<Requirement>(POS_REQUIREMENT,
PARSE_REQUIREMENT, BINARIZED_TREES_REQUIREMENT));
-
+  public static final Set<Requirement> TOKENIZE_AND_SSPLIT = Collections.unmodifiableSet(new
ArraySet<Requirement>(TOKENIZE_REQUIREMENT, SSPLIT_REQUIREMENT));
+  public static final Set<Requirement> TOKENIZE_SSPLIT_POS = Collections.unmodifiableSet(new
ArraySet<Requirement>(TOKENIZE_REQUIREMENT, SSPLIT_REQUIREMENT, POS_REQUIREMENT));
+  public static final Set<Requirement> TOKENIZE_SSPLIT_NER = Collections.unmodifiableSet(new
ArraySet<Requirement>(TOKENIZE_REQUIREMENT, SSPLIT_REQUIREMENT, NER_REQUIREMENT));
+  public static final Set<Requirement> TOKENIZE_SSPLIT_PARSE = Collections.unmodifiableSet(new
ArraySet<Requirement>(TOKENIZE_REQUIREMENT, SSPLIT_REQUIREMENT, PARSE_REQUIREMENT));
+  public static final Set<Requirement> TOKENIZE_SSPLIT_PARSE_NER = Collections.unmodifiableSet(new
ArraySet<Requirement>(TOKENIZE_REQUIREMENT, SSPLIT_REQUIREMENT, PARSE_REQUIREMENT, NER_REQUIREMENT));
+  public static final Set<Requirement> TOKENIZE_SSPLIT_POS_LEMMA = Collections.unmodifiableSet(new
ArraySet<Requirement>(TOKENIZE_REQUIREMENT, SSPLIT_REQUIREMENT, POS_REQUIREMENT, LEMMA_REQUIREMENT));
+  public static final Set<Requirement> PARSE_AND_TAG = Collections.unmodifiableSet(new
ArraySet<Requirement>(POS_REQUIREMENT, PARSE_REQUIREMENT));
+  public static final Set<Requirement> PARSE_TAG_BINARIZED_TREES = Collections.unmodifiableSet(new
ArraySet<Requirement>(POS_REQUIREMENT, PARSE_REQUIREMENT, BINARIZED_TREES_REQUIREMENT));
 }
diff --git a/src/edu/stanford/nlp/pipeline/AnnotatorImplementations.java
b/src/edu/stanford/nlp/pipeline/AnnotatorImplementations.java
index 8ee43a5..a9d08e0 100644
--- a/src/edu/stanford/nlp/pipeline/AnnotatorImplementations.java
+++ b/src/edu/stanford/nlp/pipeline/AnnotatorImplementations.java
@@ -4,7 +4,6 @@ import edu.stanford.nlp.ie.NERClassifierCombiner;
 import edu.stanford.nlp.ie.regexp.NumberSequenceClassifier;
 import edu.stanford.nlp.naturalli.NaturalLogicAnnotator;
 import edu.stanford.nlp.naturalli.OpenIE;
-import edu.stanford.nlp.util.MetaClass;
 import edu.stanford.nlp.util.PropertiesUtils;
 import edu.stanford.nlp.util.ReflectionLoading;

@@ -72,12 +71,12 @@ public class AnnotatorImplementations {
    */
   public Annotator ner(Properties properties) throws IOException {
```

```
-   List<String> models = new ArrayList<>();
+   List<String> models = new ArrayList<String>();
    String modelNames = properties.getProperty("ner.model");
    if (modelNames == null) {
      modelNames = DefaultPaths.DEFAULT_NER_THREECLASS_MODEL + "," + DefaultPaths.DEFAULT_NER_MUC_MODEL + "," +
DefaultPaths.DEFAULT_NER_CONLL_MODEL;
    }
-   if ( ! modelNames.isEmpty()) {
+   if (modelNames.length() > 0) {
      models.addAll(Arrays.asList(modelNames.split(",")));
    }
    if (models.isEmpty()) {
@@ -99,10 +98,7 @@ public class AnnotatorImplementations {

    String[] loadPaths = models.toArray(new String[models.size()]);

-   Properties combinerProperties = PropertiesUtils.extractSelectedProperties(properties,
-           NERClassifierCombiner.DEFAULT_PASS_DOWN_PROPERTIES);
-   NERClassifierCombiner nerCombiner = new NERClassifierCombiner(applyNumericClassifiers,
-           useSUTime, combinerProperties, loadPaths);
+   NERClassifierCombiner nerCombiner = new NERClassifierCombiner(applyNumericClassifiers, useSUTime, properties, loadPaths);

    int nThreads = PropertiesUtils.getInt(properties, "ner.nthreads", PropertiesUtils.getInt(properties, "nthreads", 1));
    long maxTime = PropertiesUtils.getLong(properties, "ner.maxtime", 0);
@@ -135,8 +131,8 @@ public class AnnotatorImplementations {
  /**
   * Annotate parse trees
   *
-  * @param properties Properties that control the behavior of the parser. It use "parse.x" properties.
-  * @return A ParserAnnotator
+   * @param properties
+   * @return
   */
  public Annotator parse(Properties properties) {
    String parserType = properties.getProperty("parse.type", "stanford");
@@ -166,23 +162,8 @@ public class AnnotatorImplementations {
          .CUSTOM_ANNOTATOR_PREFIX.length());
    String customClassName = properties.getProperty(property);

-   try {
-     // name + properties
-     return new MetaClass(customClassName).createInstance(customName, properties);
-   } catch (MetaClass.ConstructorNotFoundException e) {
-     try {
-       // name
-       return new MetaClass(customClassName).createInstance(customName);
-     } catch (MetaClass.ConstructorNotFoundException e2) {
-       // properties
-       try {
-         return new MetaClass(customClassName).createInstance(properties);
-       } catch (MetaClass.ConstructorNotFoundException e3) {
-         // empty arguments
-         return new MetaClass(customClassName).createInstance();
-       }
-     }
-   }
+   return ReflectionLoading.loadByReflection(customClassName, customName,
+           properties);
  }

  /**
diff --git a/src/edu/stanford/nlp/pipeline/CleanXmlAnnotator.java b/src/edu/stanford/nlp/pipeline/CleanXmlAnnotator.java
index 206e9a6..4207dfb 100644
--- a/src/edu/stanford/nlp/pipeline/CleanXmlAnnotator.java
+++ b/src/edu/stanford/nlp/pipeline/CleanXmlAnnotator.java
@@ -82,16 +82,16 @@ public class CleanXmlAnnotator implements Annotator{

  /**
```

```
   * A map of document level annotation keys (i.e. docid) along with a pattern
-  *  indicating the tag to match, and the attribute to match.
+  *  indicating the tag to match, and the attribute to match
   */
- private final CollectionValuedMap<Class, Pair<Pattern,Pattern>> docAnnotationPatterns = new CollectionValuedMap<>();
+ private CollectionValuedMap<Class, Pair<Pattern,Pattern>> docAnnotationPatterns = new CollectionValuedMap<Class, Pair<Pattern,
Pattern>>();
  public static final String DEFAULT_DOC_ANNOTATIONS_PATTERNS =
"docID=doc[id],doctype=doc[type],docsourcetype=doctype[source]";

  /**
   * A map of token level annotation keys (i.e. link, speaker) along with a pattern
-  *  indicating the tag/attribute to match (tokens that belong to the text enclosed in the specified tag will be annotated).
+  *  indicating the tag/attribute to match (tokens that belows to the text enclosed in the specified tag witll be annotated)
   */
- private final CollectionValuedMap<Class, Pair<Pattern,Pattern>> tokenAnnotationPatterns = new CollectionValuedMap<>();
+ private CollectionValuedMap<Class, Pair<Pattern,Pattern>> tokenAnnotationPatterns = new CollectionValuedMap<Class,
Pair<Pattern, Pattern>>();
  public static final String DEFAULT_TOKEN_ANNOTATIONS_PATTERNS = null;

  /**
@@ -106,10 +106,10 @@ public class CleanXmlAnnotator implements Annotator{
  private Pattern ssplitDiscardTokensMatcher = null;

  /**
-  * A map of section level annotation keys (i.e. docid) along with a pattern
-  *  indicating the tag to match, and the attribute to match.
+  * A map of section level annotation keys (i.e. docid) along with a pattern i
+  *  indicating the tag to match, and the attribute to match
   */
- private final CollectionValuedMap<Class, Pair<Pattern,Pattern>> sectionAnnotationPatterns = new CollectionValuedMap<>();
+ private CollectionValuedMap<Class, Pair<Pattern,Pattern>> sectionAnnotationPatterns = new CollectionValuedMap<Class,
Pair<Pattern, Pattern>>();
  public static final String DEFAULT_SECTION_ANNOTATIONS_PATTERNS = null;

  /**
@@ -151,8 +151,8 @@ public class CleanXmlAnnotator implements Annotator{
    dateTagMatcher = toCaseInsensitivePattern(dateTags);
  }

- private static Pattern toCaseInsensitivePattern(String tags) {
-   if (tags != null) {
+ private Pattern toCaseInsensitivePattern(String tags) {
+   if(tags != null){
     return Pattern.compile(tags, Pattern.CASE_INSENSITIVE);
    } else {
     return null;
@@ -202,7 +202,7 @@ public class CleanXmlAnnotator implements Annotator{
  }

  private static final Pattern TAG_ATTR_PATTERN = Pattern.compile("(.*)\\[(.*)\\]");
- private static void addAnnotationPatterns(CollectionValuedMap<Class, Pair<Pattern,Pattern>> annotationPatterns, String conf,
boolean attrOnly) {
+ private void addAnnotationPatterns(CollectionValuedMap<Class, Pair<Pattern,Pattern>> annotationPatterns, String conf, boolean
attrOnly) {
    String[] annoPatternStrings = conf == null ? new String[0] : conf.trim().split("\\s*,\\s*");
    for (String annoPatternString:annoPatternStrings) {
     String[] annoPattern = annoPatternString.split("\\s*=\\s*", 2);
@@ -247,7 +247,7 @@ public class CleanXmlAnnotator implements Annotator{
    return process(null, tokens);
  }

- private static String tokensToString(Annotation annotation, List<CoreLabel> tokens) {
+ private String tokensToString(Annotation annotation, List<CoreLabel> tokens) {
    if (tokens.isEmpty()) return "";
    // Try to get original text back?
    String annotationText = (annotation != null)? annotation.get(CoreAnnotations.TextAnnotation.class) : null;
@@ -262,20 +262,19 @@ public class CleanXmlAnnotator implements Annotator{
    }
```

```
  }

-  // Annotates CoreMap with information from xml tag
+  // Annotates coremap with information from xml tag

   /**
-   * Updates a CoreMap with attributes (or text context) from a tag.
-   *
+   * Updates a coremap with attributes (or text context) from a tag
    * @param annotation - Main document level annotation (from which the original text can be extracted)
-   * @param cm - CoreMap to annotate
+   * @param cm - coremap to annotate
    * @param tag - tag to process
    * @param annotationPatterns - list of annotation patterns to match
    * @param savedTokens - tokens for annotations that are text context of a tag
    * @param toAnnotate - what keys to annotate
-   * @return The set of annotations found
+   * @return
    */
-  private static Set<Class> annotateWithTag(Annotation annotation,
+  private Set<Class> annotateWithTag(Annotation annotation,
                                             CoreMap cm,
                                             XMLUtils.XMLTag tag,
                                             CollectionValuedMap<Class, Pair<Pattern,Pattern>> annotationPatterns,
diff --git a/src/edu/stanford/nlp/pipeline/DeterministicCorefAnnotator.java
b/src/edu/stanford/nlp/pipeline/DeterministicCorefAnnotator.java
index 0e9b26d..13a6749 100644
--- a/src/edu/stanford/nlp/pipeline/DeterministicCorefAnnotator.java
+++ b/src/edu/stanford/nlp/pipeline/DeterministicCorefAnnotator.java
@@ -18,13 +18,8 @@ import edu.stanford.nlp.dcoref.SieveCoreferenceSystem;
 import edu.stanford.nlp.ling.CoreAnnotations;
 import edu.stanford.nlp.ling.CoreLabel;
 import edu.stanford.nlp.dcoref.CorefCoreAnnotations;
-import edu.stanford.nlp.semgraph.SemanticGraph;
-import edu.stanford.nlp.semgraph.SemanticGraphCoreAnnotations;
-import edu.stanford.nlp.semgraph.SemanticGraphFactory;
-import edu.stanford.nlp.semgraph.SemanticGraphFactory.Mode;
 import edu.stanford.nlp.trees.Tree;
 import edu.stanford.nlp.trees.TreeCoreAnnotations;
-import edu.stanford.nlp.trees.GrammaticalStructure.Extras;
 import edu.stanford.nlp.util.ArraySet;
 import edu.stanford.nlp.util.CoreMap;
 import edu.stanford.nlp.util.Generics;
@@ -70,7 +65,7 @@ public class DeterministicCorefAnnotator implements Annotator {
   }

   @Override
-  public void annotate(Annotation annotation) {
+  public void annotate(Annotation annotation){
     try {
       List<Tree> trees = new ArrayList<Tree>();
       List<List<CoreLabel>> sentences = new ArrayList<List<CoreLabel>>();
@@ -86,9 +81,6 @@ public class DeterministicCorefAnnotator implements Annotator {
         Tree tree = sentence.get(TreeCoreAnnotations.TreeAnnotation.class);
         trees.add(tree);

-        SemanticGraph dependencies = SemanticGraphFactory.makeFromTree(tree, Mode.COLLAPSED, Extras.NONE, false, null, true);
-        sentence.set(SemanticGraphCoreAnnotations.AlternativeDependenciesAnnotation.class, dependencies);
-
         if (!hasSpeakerAnnotations) {
           // check for speaker annotations
           for (CoreLabel t:tokens) {
@@ -117,7 +109,7 @@ public class DeterministicCorefAnnotator implements Annotator {
       // add the relevant info to mentions and order them for coref
       Document document = mentionExtractor.arrange(annotation, sentences, trees, allUnprocessedMentions);
       List<List<Mention>> orderedMentions = document.getOrderedMentions();
-      if (VERBOSE) {
+      if(VERBOSE){
         for(int i = 0; i < orderedMentions.size(); i ++){
```

```
        System.err.printf("Mentions in sentence #%d:\n", i);
        for(int j = 0; j < orderedMentions.get(i).size(); j ++){
@@ -129,8 +121,62 @@ public class DeterministicCorefAnnotator implements Annotator {
      Map<Integer, CorefChain> result = corefSystem.coref(document);
      annotation.set(CorefCoreAnnotations.CorefChainAnnotation.class, result);

+     // for backward compatibility
      if(OLD_FORMAT) {
-       addObsoleteCoreferenceAnnotations(annotation, orderedMentions, result);
+       List<Pair<IntTuple, IntTuple>> links = SieveCoreferenceSystem.getLinks(result);
+
+       if(VERBOSE){
+         System.err.printf("Found %d coreference links:\n", links.size());
+         for(Pair<IntTuple, IntTuple> link: links){
+           System.err.printf("LINK (%d, %d) -> (%d, %d)\n", link.first.get(0), link.first.get(1), link.second.get(0), link.second.get(1));
+         }
+       }
+
+       //
+       // save the coref output as CorefGraphAnnotation
+       //
+
+       // cdm 2013: this block didn't seem to be doing anything needed....
+       // List<List<CoreLabel>> sents = new ArrayList<List<CoreLabel>>();
+       // for (CoreMap sentence: annotation.get(CoreAnnotations.SentencesAnnotation.class)) {
+       //   List<CoreLabel> tokens = sentence.get(CoreAnnotations.TokensAnnotation.class);
+       //   sents.add(tokens);
+       // }
+
+       // this graph is stored in CorefGraphAnnotation -- the raw links found by the coref system
+       List<Pair<IntTuple, IntTuple>> graph = new ArrayList<Pair<IntTuple,IntTuple>>();
+
+       for(Pair<IntTuple, IntTuple> link: links){
+         //
+         // Note: all offsets in the graph start at 1 (not at 0!)
+         //       we do this for consistency reasons, as indices for syntactic dependencies start at 1
+         //
+         int srcSent = link.first.get(0);
+         int srcTok = orderedMentions.get(srcSent - 1).get(link.first.get(1)-1).headIndex + 1;
+         int dstSent = link.second.get(0);
+         int dstTok = orderedMentions.get(dstSent - 1).get(link.second.get(1)-1).headIndex + 1;
+         IntTuple dst = new IntTuple(2);
+         dst.set(0, dstSent);
+         dst.set(1, dstTok);
+         IntTuple src = new IntTuple(2);
+         src.set(0, srcSent);
+         src.set(1, srcTok);
+         graph.add(new Pair<IntTuple, IntTuple>(src, dst));
+       }
+       annotation.set(CorefCoreAnnotations.CorefGraphAnnotation.class, graph);
+
+       for (CorefChain corefChain : result.values()) {
+         if(corefChain.getMentionsInTextualOrder().size() < 2) continue;
+         Set<CoreLabel> coreferentTokens = Generics.newHashSet();
+         for (CorefMention mention : corefChain.getMentionsInTextualOrder()) {
+           CoreMap sentence = annotation.get(CoreAnnotations.SentencesAnnotation.class).get(mention.sentNum - 1);
+           CoreLabel token = sentence.get(CoreAnnotations.TokensAnnotation.class).get(mention.headIndex - 1);
+           coreferentTokens.add(token);
+         }
+         for (CoreLabel token : coreferentTokens) {
+           token.set(CorefCoreAnnotations.CorefClusterAnnotation.class, coreferentTokens);
+         }
+       }
      }
    } catch (RuntimeException e) {
      throw e;
@@ -139,75 +185,14 @@ public class DeterministicCorefAnnotator implements Annotator {
    }
  }
```

```
- // for backward compatibility with a few old things
- // TODO: Aim to get rid of this entirely
- private static void addObsoleteCoreferenceAnnotations(Annotation annotation, List<List<Mention>> orderedMentions,
-                                        Map<Integer, CorefChain> result) {
-   List<Pair<IntTuple, IntTuple>> links = SieveCoreferenceSystem.getLinks(result);
-
-   if(VERBOSE){
-     System.err.printf("Found %d coreference links:\n", links.size());
-     for(Pair<IntTuple, IntTuple> link: links){
-       System.err.printf("LINK (%d, %d) -> (%d, %d)\n", link.first.get(0), link.first.get(1), link.second.get(0), link.second.get(1));
-     }
-   }
-
-   //
-   // save the coref output as CorefGraphAnnotation
-   //
-
-   // cdm 2013: this block didn't seem to be doing anything needed....
-   // List<List<CoreLabel>> sents = new ArrayList<List<CoreLabel>>();
-   // for (CoreMap sentence: annotation.get(CoreAnnotations.SentencesAnnotation.class)) {
-   //   List<CoreLabel> tokens = sentence.get(CoreAnnotations.TokensAnnotation.class);
-   //   sents.add(tokens);
-   // }
-
-   // this graph is stored in CorefGraphAnnotation -- the raw links found by the coref system
-   List<Pair<IntTuple, IntTuple>> graph = new ArrayList<Pair<IntTuple,IntTuple>>();
-
-   for(Pair<IntTuple, IntTuple> link: links){
-     //
-     // Note: all offsets in the graph start at 1 (not at 0!)
-     //       we do this for consistency reasons, as indices for syntactic dependencies start at 1
-     //
-     int srcSent = link.first.get(0);
-     int srcTok = orderedMentions.get(srcSent - 1).get(link.first.get(1)-1).headIndex + 1;
-     int dstSent = link.second.get(0);
-     int dstTok = orderedMentions.get(dstSent - 1).get(link.second.get(1)-1).headIndex + 1;
-     IntTuple dst = new IntTuple(2);
-     dst.set(0, dstSent);
-     dst.set(1, dstTok);
-     IntTuple src = new IntTuple(2);
-     src.set(0, srcSent);
-     src.set(1, srcTok);
-     graph.add(new Pair<>(src, dst));
-   }
-   annotation.set(CorefCoreAnnotations.CorefGraphAnnotation.class, graph);
-
-   for (CorefChain corefChain : result.values()) {
-     if(corefChain.getMentionsInTextualOrder().size() < 2) continue;
-     Set<CoreLabel> coreferentTokens = Generics.newHashSet();
-     for (CorefMention mention : corefChain.getMentionsInTextualOrder()) {
-       CoreMap sentence = annotation.get(CoreAnnotations.SentencesAnnotation.class).get(mention.sentNum - 1);
-       CoreLabel token = sentence.get(CoreAnnotations.TokensAnnotation.class).get(mention.headIndex - 1);
-       coreferentTokens.add(token);
-     }
-     for (CoreLabel token : coreferentTokens) {
-       token.set(CorefCoreAnnotations.CorefClusterAnnotation.class, coreferentTokens);
-     }
-   }
- }
-

   @Override
   public Set<Requirement> requires() {
-    return new ArraySet<>(TOKENIZE_REQUIREMENT, SSPLIT_REQUIREMENT, POS_REQUIREMENT, NER_REQUIREMENT,
PARSE_REQUIREMENT);
+    return new ArraySet<Requirement>(TOKENIZE_REQUIREMENT, SSPLIT_REQUIREMENT, POS_REQUIREMENT, NER_REQUIREMENT,
PARSE_REQUIREMENT);
   }
```

```
  @Override
  public Set<Requirement> requirementsSatisfied() {
    return Collections.singleton(DETERMINISTIC_COREF_REQUIREMENT);
  }
-
 }
diff --git a/src/edu/stanford/nlp/pipeline/StanfordCoreNLP.java b/src/edu/stanford/nlp/pipeline/StanfordCoreNLP.java
index c0442ab..9acccc7 100644
--- a/src/edu/stanford/nlp/pipeline/StanfordCoreNLP.java
+++ b/src/edu/stanford/nlp/pipeline/StanfordCoreNLP.java
@@ -520,6 +520,16 @@ public class StanfordCoreNLP extends AnnotationPipeline {
    }
  }

+  public void xmlPrint(Annotation annotation, OutputStream os, String filename) throws IOException {
+    try {
+      Class clazz = Class.forName("edu.stanford.nlp.pipeline.XMLOutputter");
+      Method method = clazz.getMethod("xmlPrint", Annotation.class, OutputStream.class, StanfordCoreNLP.class, String.class);
+      method.invoke(null, annotation, os, this, filename);
+    } catch (NoSuchMethodException | IllegalAccessException | ClassNotFoundException | InvocationTargetException e) {
+      throw new RuntimeException(e);
+    }
+  }
+
  //
  // runtime, shell-specific, and help menu methods
  //
@@ -887,7 +897,7 @@ public class StanfordCoreNLP extends AnnotationPipeline {
          switch (outputFormat) {
          case XML: {
            OutputStream fos = new BufferedOutputStream(new FileOutputStream(finalOutputFilename));
-           xmlPrint(annotation, fos);
+           xmlPrint(annotation, fos, finalOutputFilename);
            fos.close();
            break;
          }
@@ -934,8 +944,10 @@ public class StanfordCoreNLP extends AnnotationPipeline {

        endTrack("Processing file " + file.getAbsolutePath() + " ... writing to " + finalOutputFilename);

-      } catch (IOException e) {
-        throw new RuntimeIOException(e);
+      } catch (Exception e) {
+            System.err.println("IGNORING ERROR:");
+            e.printStackTrace(System.err);
+        // throw new RuntimeIOException(e);
       }
     });
   }
diff --git a/src/edu/stanford/nlp/pipeline/XMLOutputter.java b/src/edu/stanford/nlp/pipeline/XMLOutputter.java
index 4423bfb..8582f37 100644
--- a/src/edu/stanford/nlp/pipeline/XMLOutputter.java
+++ b/src/edu/stanford/nlp/pipeline/XMLOutputter.java
@@ -2,11 +2,17 @@ package edu.stanford.nlp.pipeline;

 import java.io.IOException;
 import java.io.OutputStream;
+import java.io.FileOutputStream;
+import java.io.BufferedOutputStream;
 import java.io.PrintWriter;
 import java.io.StringWriter;
 import java.util.List;
+import java.util.ArrayList;
 import java.util.Map;
+import java.util.HashMap;
+import java.util.Set;

+import org.decidingforce.DFClausIE;
```

```
 import edu.stanford.nlp.dcoref.CorefChain;
 import edu.stanford.nlp.dcoref.CorefCoreAnnotations;
 import edu.stanford.nlp.ie.machinereading.structure.EntityMention;
@@ -30,7 +36,9 @@ import edu.stanford.nlp.semgraph.SemanticGraphCoreAnnotations;
 import edu.stanford.nlp.semgraph.SemanticGraphEdge;
 import edu.stanford.nlp.util.CoreMap;
 import edu.stanford.nlp.util.Pair;
+import edu.stanford.nlp.util.IntPair;
 import edu.stanford.nlp.util.StringUtils;
+import org.codehaus.jackson.map.ObjectMapper;
 import nu.xom.*;


@@ -44,18 +52,38 @@ public class XMLOutputter extends AnnotationOutputter {
   // the namespace is set in the XSLT file
   private static final String NAMESPACE_URI = null;
   private static final String STYLESHEET_NAME = "CoreNLP-to-HTML.xsl";
+  public String associated_filename;
+  public String prefix;
+  public HashMap<String,BufferedOutputStream> filemap;

   public XMLOutputter() {}

   /** {@inheritDoc} */
   @Override
   public void print(Annotation annotation, OutputStream os, Options options) throws IOException {
-    Document xmlDoc = annotationToDoc(annotation, options);
-    Serializer ser = new Serializer(os, options.encoding);
-    ser.setIndent(2);
-    ser.setMaxLength(0);
-    ser.write(xmlDoc);
-    ser.flush();
+    // Document xmlDoc = annotationToDoc(annotation, options);
+    // Serializer ser = new Serializer(os, options.encoding);
+    // ser.setIndent(2);
+    // ser.setMaxLength(0);
+    // ser.write(xmlDoc);
+    // ser.flush();
+    filemap = new HashMap<String,BufferedOutputStream>();
+    ArrayList<Map> al = annotationToSVO(annotation, options, prefix, filemap);
+    for (String k: filemap.keySet()) {
+      filemap.get(k).close();
+    }
+    if (al == null) return;
+    ObjectMapper mapper = new ObjectMapper();
+    for (Map svo : al) {
+      try {
+        System.err.println("Writing: " + mapper.writeValueAsString(svo));
+        os.write(mapper.writeValueAsString(svo).getBytes());
+        os.write("\n".getBytes());
+      } catch (IOException e) {
+        e.printStackTrace();
+      }
+    }
+    os.flush();
   }

   public static void xmlPrint(Annotation annotation, OutputStream os) throws IOException {
@@ -66,6 +94,13 @@ public class XMLOutputter extends AnnotationOutputter {
     new XMLOutputter().print(annotation, os, pipeline);
   }

+  public static void xmlPrint(Annotation annotation, OutputStream os, StanfordCoreNLP pipeline, String filename) throws IOException {
+    XMLOutputter o = new XMLOutputter();
+    o.associated_filename = filename;
+    o.prefix = filename.substring(0, filename.lastIndexOf('.'));
+    o.print(annotation, os, pipeline);
+  }
+
```

```
  public static void xmlPrint(Annotation annotation, OutputStream os, Options options) throws IOException {
    new XMLOutputter().print(annotation, os, options);
  }
@@ -78,9 +113,175 @@ public class XMLOutputter extends AnnotationOutputter {
    return annotationToDoc(annotation, options);
  }

+  public static void writeTUA(CoreLabel token, String filename, String type, String word, String orig,
HashMap<String,BufferedOutputStream> filemap) {
+    String[] parts = type.split("\\|");
+    String type_id = parts[0] + '-' + parts[1];
+    filename += "-" + type_id + ".tua";
+    try {
+      BufferedOutputStream os;
+      if (!filemap.containsKey(filename)) {
+        os = new BufferedOutputStream(new FileOutputStream(filename));
+        filemap.put(filename, os);
+      }
+      os = filemap.get(filename);
+      token.set(CoreAnnotations.DFTypeAnnotation.class, filename);
+      os.write(word.getBytes());
+      os.write(" ".getBytes());
+    } catch (IOException e) {
+      System.err.println("Problem writing TUA file: " + filename);
+      e.printStackTrace();
+    }
+  }
+
   /**
    * Converts the given annotation to an XML document using the specified options
    */
+  public static ArrayList<Map> annotationToSVO(Annotation annotation, Options options, String filename,
HashMap<String,BufferedOutputStream> filemap) {
+    ArrayList<Map> result = new ArrayList<Map>();
+
+    Map<Integer, CorefChain> corefChains =
+      annotation.get(CorefCoreAnnotations.CorefChainAnnotation.class);
+    if (corefChains != null) {
+      dcorefTransform(annotation, corefChains);
+    }
+    String text = "ERROR: there should be text here";
+    if(annotation.get(CoreAnnotations.SentencesAnnotation.class) != null){
+      int sentCount = 1;
+      for (CoreMap sentence: annotation.get(CoreAnnotations.SentencesAnnotation.class)) {
+        text = "";
+        Integer lineNumber = sentence.get(CoreAnnotations.LineNumberAnnotation.class);
+        Tree tree = sentence.get(TreeCoreAnnotations.TreeAnnotation.class);
+        SemanticGraph basicDependencies = sentence.get(SemanticGraphCoreAnnotations.BasicDependenciesAnnotation.class);
+        List<CoreLabel> tokens = sentence.get(CoreAnnotations.TokensAnnotation.class);
+        // System.out.print("Dcoref: ");
+        StringBuffer s = new StringBuffer();
+
+        for(int j = 0; j < tokens.size(); j ++) {
+          int tokNum = j + 1;
+          // if (corefChains != null) {
+          //   addWordInfo(tokens.get(j), tokNum, sentCount, corefChains);
+          // }
+          CoreLabel token = tokens.get(j);
+          String word = token.get(CoreAnnotations.TextAnnotation.class);
+          String orig = token.get(CoreAnnotations.OriginalTextAnnotation.class);
+          String type = token.get(CoreAnnotations.DFTypeIDAnnotation.class);
+          if (word != "") {
+            s.append(word + " ");
+            if (null != type) writeTUA(token, filename, type, word, orig, filemap);
+          }
+
+          // System.out.print(word + " ");
+        }
+        System.out.println();
```

```
+       text = s.toString();
+       sentCount ++;
+
+       try {
+         if (tree != null && basicDependencies != null) {
+           // String text = sentence.get(CoreAnnotations.TextAnnotation.class);
+           final DFClausIE df = new DFClausIE(text, basicDependencies, tree);
+           result.addAll(df.run());
+         }
+       } catch (IOException e) {
+         System.err.println("There was a problem with DF");
+       }
+     }
+   }
+
+   return result;
+ }
+
+
+ private static int countMentions(CorefChain cluster) {
+   int count = 0;
+   for(IntPair mid: cluster.getMentionMap().keySet()) {
+     count += cluster.getMentionMap().get(mid).size();
+   }
+   return count;
+ }
+
+ private static void dcorefTransform(Annotation annotation, Map<Integer, CorefChain> corefChains) {
+   List<CoreMap> sentences = annotation.get(CoreAnnotations.SentencesAnnotation.class);
+   int cid = 0;
+   for (CorefChain cluster : corefChains.values()) {
+     // System.out.println(cid + " " + countMentions(cluster));
+     // each mention saved on one line
+     Map<IntPair, Set<CorefChain.CorefMention>> mentionMap = cluster.getMentionMap();
+     for(IntPair mid: mentionMap.keySet()) {
+       // all mentions with the same head
+       Set<CorefChain.CorefMention> mentions = mentionMap.get(mid);
+       for(CorefChain.CorefMention mention: mentions) {
+         int sentNum = mention.sentNum - 1;
+         int headIndex = mention.headIndex - 1;
+         int startIndex = mention.startIndex - 1;
+         int endIndex = mention.endIndex - 1;
+         System.out.print(" " + sentNum);
+         System.out.print(" " + headIndex);
+         System.out.print(" " + startIndex);
+         System.out.print(" " + endIndex);
+         System.out.println();
+
+         CoreMap sentence = sentences.get(sentNum);
+         List<CoreLabel> tokens = sentence.get(CoreAnnotations.TokensAnnotation.class);
+
+         for (int i = startIndex; i < endIndex; i++) {
+           CoreLabel token = tokens.get(i);
+           if (token.get(CoreAnnotations.OriginalTextAnnotation.class) == null) {
+             token.set(CoreAnnotations.OriginalTextAnnotation.class, token.get(CoreAnnotations.TextAnnotation.class));
+           }
+           if (i == headIndex) {
+             CorefChain.CorefMention rep = cluster.getRepresentativeMention();
+             token.set(CoreAnnotations.TextAnnotation.class, rep.mentionSpan);
+             System.out.format("head token: %s\n", token);
+           } else {
+             // token.set(CoreAnnotations.TextAnnotation.class, "DELETED");
+             token.set(CoreAnnotations.TextAnnotation.class, "");
+             // token.remove();
+             System.out.format("token: %s\n", token);
+           }
+         }
+         // one mention per line
+         // System.out.print(" " + mention.position.length());
```

```
+         // System.out.print(mid.getSource() + " " + mid.getTarget());
+         // if(mention == cluster.getRepresentativeMention()) System.out.print(" " + 1);
+         // else System.out.print(" " + 0);
+
+         // System.out.print(" " + mention.mentionType);
+         // System.out.print(" " + mention.number);
+         // System.out.print(" " + mention.gender);
+         // System.out.print(" " + mention.animacy);
+         // System.out.print(" " + mention.corefClusterID);
+         // System.out.print(" " + mention.mentionID);
+         // System.out.print(" {");
+         // for(int i = 0; i < mention.position.length(); i ++)
+         //   System.out.print(" " + mention.position.get(i));
+         // System.out.print("} | " + mention.mentionSpan);
+         System.out.println();
+       }
+     }
+     cid++;
+   }
+ }
+
+ private static void addWordInfo(CoreMap token, int id, int sentCount, Map<Integer, CorefChain> corefChains) {
+   for (CorefChain chain : corefChains.values()) {
+          // System.err.println("chain: " + chain.getMentionMap().get(sentCount));
+          // System.err.println("chain: " + sentCount + " " + id);
+          Set<CorefChain.CorefMention> set = chain.getMentionsWithSameHead(sentCount, id);
+          if (set != null && !set.isEmpty()){
+                CorefChain.CorefMention rep = chain.getRepresentativeMention();
+                String original = token.get(CoreAnnotations.TextAnnotation.class);
+                token.set(CoreAnnotations.OriginalTextAnnotation.class, original);
+                System.out.format("token: %s\trep.mentionSpan: %s\tchain: %s\n", token, rep.mentionSpan, chain);
+                token.set(CoreAnnotations.TextAnnotation.class, "{"+rep.mentionSpan+"}");
+                for (CorefChain.CorefMention m : set) {
+                  System.out.format("mention: m.headIndex: %s\tm.startIndex: %s\tm.endIndex: %s\n", m.headIndex, m.startIndex,
m.endIndex);
+                      // token.set(CoreAnnotations.TextAnnotation.class, rep.mentionSpan);
+                }
+         }
+   }
+ }
+
  public static Document annotationToDoc(Annotation annotation, Options options) {
    //
    // create the XML document with the root node pointing to the namespace URL
@@ -88,7 +289,7 @@ public class XMLOutputter extends AnnotationOutputter {
    Element root = new Element("root", NAMESPACE_URI);
    Document xmlDoc = new Document(root);
    ProcessingInstruction pi = new ProcessingInstruction("xml-stylesheet",
-         "href=\"" + STYLESHEET_NAME + "\" type=\"text/xsl\"");
+        "href=\"" + STYLESHEET_NAME + "\" type=\"text/xsl\"");
    xmlDoc.insertChild(pi, 0);
    Element docElem = new Element("document", NAMESPACE_URI);
    root.appendChild(docElem);
@@ -107,6 +308,9 @@ public class XMLOutputter extends AnnotationOutputter {
    Element sentencesElem = new Element("sentences", NAMESPACE_URI);
    docElem.appendChild(sentencesElem);

+    Map<Integer, CorefChain> corefChains =
+      annotation.get(CorefCoreAnnotations.CorefChainAnnotation.class);
+
    //
    // save the info for each sentence in this doc
    //
@@ -119,16 +323,17 @@ public class XMLOutputter extends AnnotationOutputter {
        if (lineNumber != null) {
          sentElem.addAttribute(new Attribute("line", Integer.toString(lineNumber)));
        }
-       sentCount ++;
```

```
        // add the word table with all token-level annotations
        Element wordTable = new Element("tokens", NAMESPACE_URI);
        List<CoreLabel> tokens = sentence.get(CoreAnnotations.TokensAnnotation.class);
-       for(int j = 0; j < tokens.size(); j ++){
+       for(int j = 0; j < tokens.size(); j ++) {
          Element wordInfo = new Element("token", NAMESPACE_URI);
-         addWordInfo(wordInfo, tokens.get(j), j + 1, NAMESPACE_URI);
+         int tokNum = j + 1;
+         addWordInfo(wordInfo, tokens.get(j), tokNum, sentCount, corefChains, NAMESPACE_URI);
          wordTable.appendChild(wordInfo);
        }
+       sentCount ++;
        sentElem.appendChild(wordTable);

        // add tree info
@@ -142,29 +347,36 @@ public class XMLOutputter extends AnnotationOutputter {
        }

        SemanticGraph basicDependencies = sentence.get(SemanticGraphCoreAnnotations.BasicDependenciesAnnotation.class);
+       try {
+           String text = sentence.get(CoreAnnotations.TextAnnotation.class);
+           final DFClausIE df = new DFClausIE(text, basicDependencies, tree);
+       df.run();
+       } catch (IOException e) {
+           System.err.println("There was a problem with DF");
+       }

-       if (basicDependencies != null) {
-         // add the dependencies for this sentence
-         Element depInfo = buildDependencyTreeInfo("basic-dependencies",
sentence.get(SemanticGraphCoreAnnotations.BasicDependenciesAnnotation.class), tokens, NAMESPACE_URI);
-         if (depInfo != null) {
-           sentElem.appendChild(depInfo);
-         }
+       // if (basicDependencies != null) {
+       //   // add the dependencies for this sentence
+       //   Element depInfo = buildDependencyTreeInfo("basic-dependencies",
sentence.get(SemanticGraphCoreAnnotations.BasicDependenciesAnnotation.class), tokens, NAMESPACE_URI);
+       //   if (depInfo != null) {
+       //     sentElem.appendChild(depInfo);
+       //   }

-         depInfo = buildDependencyTreeInfo("collapsed-dependencies",
sentence.get(SemanticGraphCoreAnnotations.CollapsedDependenciesAnnotation.class), tokens, NAMESPACE_URI);
-         if (depInfo != null) {
-           sentElem.appendChild(depInfo);
-         }
+       //   depInfo = buildDependencyTreeInfo("collapsed-dependencies",
sentence.get(SemanticGraphCoreAnnotations.CollapsedDependenciesAnnotation.class), tokens, NAMESPACE_URI);
+       //   if (depInfo != null) {
+       //     sentElem.appendChild(depInfo);
+       //   }

-         depInfo = buildDependencyTreeInfo("collapsed-ccprocessed-dependencies",
sentence.get(SemanticGraphCoreAnnotations.CollapsedCCProcessedDependenciesAnnotation.class), tokens, NAMESPACE_URI);
-         if (depInfo != null) {
-           sentElem.appendChild(depInfo);
-         }
-       }
+       //   depInfo = buildDependencyTreeInfo("collapsed-ccprocessed-dependencies",
sentence.get(SemanticGraphCoreAnnotations.CollapsedCCProcessedDependenciesAnnotation.class), tokens, NAMESPACE_URI);
+       //   if (depInfo != null) {
+       //     sentElem.appendChild(depInfo);
+       //   }
+       // }

        // add the MR entities and relations
        List<EntityMention> entities = sentence.get(MachineReadingAnnotations.EntityMentionsAnnotation.class);
        List<RelationMention> relations = sentence.get(MachineReadingAnnotations.RelationMentionsAnnotation.class);
```

```
-       if (entities != null && ! entities.isEmpty()){
+       if (entities != null && entities.size() > 0){
         Element mrElem = new Element("MachineReading", NAMESPACE_URI);
         Element entElem = new Element("entities", NAMESPACE_URI);
         addEntities(entities, entElem, NAMESPACE_URI);
@@ -179,17 +391,19 @@ public class XMLOutputter extends AnnotationOutputter {
         sentElem.appendChild(mrElem);
       }

+
       /**
        * Adds sentiment as an attribute of this sentence.
        */
-      Tree sentimentTree = sentence.get(SentimentCoreAnnotations.SentimentAnnotatedTree.class);
+      Tree sentimentTree = sentence.get(SentimentCoreAnnotations.AnnotatedTree.class);
       if (sentimentTree != null) {
         int sentiment = RNNCoreAnnotations.getPredictedClass(sentimentTree);
         sentElem.addAttribute(new Attribute("sentimentValue", Integer.toString(sentiment)));
-        String sentimentClass = sentence.get(SentimentCoreAnnotations.SentimentClass.class);
+        String sentimentClass = sentence.get(SentimentCoreAnnotations.ClassName.class);
         sentElem.addAttribute(new Attribute("sentiment", sentimentClass.replaceAll(" ", "")));
       }

+
       // add the sentence to the root
       sentencesElem.appendChild(sentElem);
     }
@@ -198,14 +412,14 @@ public class XMLOutputter extends AnnotationOutputter {
     //
     // add the coref graph
     //
-    Map<Integer, CorefChain> corefChains =
-        annotation.get(CorefCoreAnnotations.CorefChainAnnotation.class);
-    if (corefChains != null) {
-      List<CoreMap> sentences = annotation.get(CoreAnnotations.SentencesAnnotation.class);
-      Element corefInfo = new Element("coreference", NAMESPACE_URI);
-      if (addCorefGraphInfo(options, corefInfo, sentences, corefChains, NAMESPACE_URI))
-        docElem.appendChild(corefInfo);
-    }
+    // Map<Integer, CorefChain> corefChains =
+    //   annotation.get(CorefCoreAnnotations.CorefChainAnnotation.class);
+    // if (corefChains != null) {
+    //   List<CoreMap> sentences = annotation.get(CoreAnnotations.SentencesAnnotation.class);
+    //   Element corefInfo = new Element("coreference", NAMESPACE_URI);
+    //   if (addCorefGraphInfo(options, corefInfo, sentences, corefChains, NAMESPACE_URI))
+    //     docElem.appendChild(corefInfo);
+    // }

     //
     // save any document-level annotations here
@@ -238,7 +452,7 @@ public class XMLOutputter extends AnnotationOutputter {
       int target = root.index();
       String sourceWord = "ROOT";
       String targetWord = tokens.get(target - 1).word();
-      final boolean isExtra = false;
+      boolean isExtra = false;

       addDependencyInfo(depInfo, rel, isExtra, source, sourceWord, null, target, targetWord, null, curNS);
     }
@@ -287,7 +501,7 @@ public class XMLOutputter extends AnnotationOutputter {
   }

   /**
-   * Generates the XML content for MachineReading entities.
+   * Generates the XML content for MachineReading entities
    */
   private static void addEntities(List<EntityMention> entities, Element top, String curNS) {
     for (EntityMention e: entities) {
@@ -297,11 +511,11 @@ public class XMLOutputter extends AnnotationOutputter {
```

```
  }

  /**
-  * Generates the XML content for MachineReading relations.
+  * Generates the XML content for MachineReading relations
   */
  private static void addRelations(List<RelationMention> relations, Element top, String curNS, double beam){
-   for (RelationMention r: relations){
-     if (r.printableObject(beam)) {
+   for(RelationMention r: relations){
+     if(r.printableObject(beam)) {
       Element re = toXML(r, curNS);
       top.appendChild(re);
     }
@@ -309,7 +523,7 @@ public class XMLOutputter extends AnnotationOutputter {
  }

  /**
-  * Generates the XML content for the coreference chain object.
+  * Generates the XML content for the coreference chain object
   */
  private static boolean addCorefGraphInfo
    (Options options, Element corefInfo, List<CoreMap> sentences, Map<Integer, CorefChain> corefChains, String curNS)
@@ -369,10 +583,41 @@ public class XMLOutputter extends AnnotationOutputter {
    chainElem.appendChild(mentionElem);
  }

- private static void addWordInfo(Element wordInfo, CoreMap token, int id, String curNS) {
+ private static void addWordInfo(Element wordInfo, CoreMap token, int id, int sentCount, Map<Integer, CorefChain> corefChains, String
curNS) {
    // store the position of this word in the sentence
    wordInfo.addAttribute(new Attribute("id", Integer.toString(id)));
+   // wordInfo.addAttribute(new Attribute("foomen", token.get(CorefCoreAnnotations.CorefChainAnnotation.class)));
+   // token.get(CorefCoreAnnotations.CorefChainAnnotation.class)
+   String dfType = token.get(CoreAnnotations.DFTypeIDAnnotation.class);
+   if (dfType != null) {
+       wordInfo.addAttribute(new Attribute("df-type-id", dfType));
+   }
+   for (CorefChain chain : corefChains.values()) {
+       // System.err.println(chain.getMentionMap().get(sentCount));
+       // System.err.println(sentCount + " " + tokNum);
+       Set<CorefChain.CorefMention> set = chain.getMentionsWithSameHead(sentCount, id);
+       if (set != null && !set.isEmpty()){
+           CorefChain.CorefMention rep = chain.getRepresentativeMention();
+           for (CorefChain.CorefMention m : set) {
+               String original = token.get(CoreAnnotations.TextAnnotation.class);
+               token.set(CoreAnnotations.OriginalTextAnnotation.class, original);
+               setSingleElement(wordInfo, "originalWord", curNS, original);
+               token.set(CoreAnnotations.TextAnnotation.class, rep.mentionSpan);
+           }
+       }
+   }

+ //   for (Map.Entry<Integer,CorefChain> entry : corefChains.entrySet()){
+ //       Integer i = entry.getKey();
+ //       if (i == sentCount) {
+ //           CorefChain chain = corefChains.get(i);
+ //           Set<CorefChain.CorefMention> mentions = chain.getMentionsWithSameHead(sentCount, id);
+ //           CorefChain.CorefMention rep = chain.getRepresentativeMention();
+ //           setSingleElement(wordInfo, "mention", curNS, rep.mentionSpan);
+ //       }
+ //   }
+ // // public Set<CorefMention> getMentionsWithSameHead(int sentenceNumber, int headIndex) {
+ // // public CorefMention getRepresentativeMention() { return representative; }
    setSingleElement(wordInfo, "word", curNS, token.get(CoreAnnotations.TextAnnotation.class));
    setSingleElement(wordInfo, "lemma", curNS, token.get(CoreAnnotations.LemmaAnnotation.class));

@@ -417,12 +662,6 @@ public class XMLOutputter extends AnnotationOutputter {
     wordInfo.appendChild(cur);
```

```
    }

-   if (token.containsKey(SentimentCoreAnnotations.SentimentClass.class)) {
-     Element cur = new Element("sentiment", curNS);
-     cur.appendChild(token.get(SentimentCoreAnnotations.SentimentClass.class));
-     wordInfo.appendChild(cur);
-   }
-
 //    IntTuple corefDest;
 //    if((corefDest = label.get(CorefDestAnnotation.class)) != null){
 //      Element cur = new Element("coref", curNS);
@@ -489,9 +728,9 @@ public class XMLOutputter extends AnnotationOutputter {
      top.appendChild(relation.getSubType());
    }

-   List<EntityMention> mentions = relation.getEntityMentionArgs();
+   List<EntityMention> ents = relation.getEntityMentionArgs();
    Element args = new Element("arguments", curNS);
-   for (EntityMention e : mentions) {
+   for (EntityMention e : ents) {
      args.appendChild(toXML(e, curNS));
    }
    top.appendChild(args);
@@ -518,5 +757,6 @@ public class XMLOutputter extends AnnotationOutputter {
    return probs;
  }

-}

+
+}
diff --git a/src/org/decidingforce/DFClausIE.java b/src/org/decidingforce/DFClausIE.java
new file mode 100644
index 0000000..aeb213d
--- /dev/null
+++ b/src/org/decidingforce/DFClausIE.java
@@ -0,0 +1,352 @@
+package org.decidingforce;
+
+import java.io.DataInput;
+import java.io.DataInputStream;
+import java.io.FileInputStream;
+import java.io.FileOutputStream;
+import java.io.BufferedOutputStream;
+import java.io.File;
+import java.io.FileNotFoundException;
+import java.io.IOException;
+import java.io.InputStream;
+import java.io.OutputStream;
+import java.io.PrintStream;
+import java.io.StringReader;
+import java.util.ArrayList;
+import java.util.Collection;
+import java.util.HashMap;
+import java.util.List;
+import java.util.Map;
+
+import clausie.Constituent.Flag;
+import clausie.JavaUtils.MapUtil;
+import clausie.Clause;
+import clausie.Proposition;
+import clausie.Options;
+import clausie.IndexedConstituent;
+import clausie.DefaultPropositionGenerator;
+import clausie.XcompConstituent;
+import clausie.TextConstituent;
+import clausie.Constituent;
+import clausie.PropositionGenerator;
+import clausie.ClauseDetector;
```

```
+import clausie.ProcessConjunctions;
+import joptsimple.OptionException;
+import joptsimple.OptionParser;
+import joptsimple.OptionSet;
+import edu.stanford.nlp.io.EncodingPrintWriter.out;
+import edu.stanford.nlp.ling.CoreLabel;
+import edu.stanford.nlp.ling.IndexedWord;
+import edu.stanford.nlp.pipeline.ParserAnnotatorUtils;
+import edu.stanford.nlp.process.CoreLabelTokenFactory;
+import edu.stanford.nlp.process.PTBTokenizer;
+import edu.stanford.nlp.semgraph.SemanticGraph;
+import edu.stanford.nlp.semgraph.SemanticGraphFactory;
+import edu.stanford.nlp.trees.Tree;
+import edu.stanford.nlp.util.ScoredObject;
+import org.codehaus.jackson.map.ObjectMapper;
+
+public class DFClausIE {
+               Tree depTree;
+               SemanticGraph semanticGraph;
+               String text;
+               List<ScoredObject<Tree>> trees;
+
+      List<Clause> clauses = new ArrayList<Clause>();
+
+      int k = 10;
+
+      double bestScore;
+      Tree bestDT;
+      SemanticGraph bestSemanticGraph;
+      List<Clause> bestClauses;
+
+      List<Proposition> propositions = new ArrayList<Proposition>();
+      Map<Proposition, Double> scoredPropositions = new HashMap<Proposition, Double>();
+
+      PropositionGenerator propositionGenerator;
+
+      Options options;
+
+      // Indicates if the clause processed comes from an xcomp constituent of the
+      // original sentence
+      boolean xcomp = false;
+
+      // -- construction
+      // ------------------------------------------------------------------------
+
+      public DFClausIE(Options options) {
+            this.options = options;
+            this.propositionGenerator = new DefaultPropositionGenerator(this.options);
+      }
+
+      public DFClausIE() {
+            this(new Options());
+      }
+
+
+      public DFClausIE(String text, SemanticGraph sg, Tree t) throws IOException {
+            this(new Options("resources/clausie.conf"));
+            this.text = text;
+            this.semanticGraph = sg;
+            this.depTree = t;
+      }
+
+      // -- misc method
+      // ------------------------------------------------------------------------
+
+      public Options getOptions() {
+            return options;
+      }
+
```

```
+	public void clear() {
+		semanticGraph = null;
+		depTree = null;
+		clauses.clear();
+		propositions.clear();
+	}
+
+	/** Returns the constituent tree for the sentence. */
+	public Tree getDepTree() {
+		return depTree;
+	}
+
+	/** Returns the dependency tree for the sentence. */
+	public SemanticGraph getSemanticGraph() {
+		return semanticGraph;
+	}
+
+	// -- clause detection
+	// ---------------------------------------------------------------------
+
+	/** Detects clauses in the sentence. */
+	public void detectClauses() {
+	  // System.err.println("ClauseDetector.detectClauses:" + this.options + this.semanticGraph + this.depTree + this.clauses);
+	  if (this.semanticGraph != null || this.depTree != null) {
+	  ClauseDetector.detectClauses(this.options, this.semanticGraph, this.depTree, this.clauses);
+	   }
+	}
+
+	/** Returns clauses in the sentence. */
+	public List<Clause> getClauses() {
+		return clauses;
+	}
+
+	// -- proposition generation
+	// ---------------------------------------------------------------------
+
+	/** Generates propositions from the clauses in the sentence. */
+	public void generatePropositions() {
+		propositions.clear();
+
+		// holds alternative options for each constituents (obtained by
+		// processing coordinated conjunctions and xcomps)
+		final List<List<Constituent>> constituents = new ArrayList<List<Constituent>>();
+
+		// which of the constituents are required?
+		final List<Flag> flags = new ArrayList<Flag>();
+		final List<Boolean> include = new ArrayList<Boolean>();
+
+		// holds all valid combination of constituents for which a proposition
+		// is to be generated
+		final List<List<Boolean>> includeConstituents = new ArrayList<List<Boolean>>();
+
+		// let's start
+		for (Clause clause : clauses) {
+			// process coordinating conjunctions
+			constituents.clear();
+			for (int i = 0; i < clause.getConstituents().size(); i++) {
+				// if(xcomp && clause.subject == i) continue; //An xcomp does
+				// not have an internal subject so should not be processed here
+				Constituent constituent = clause.getConstituents().get(i);
+				List<Constituent> alternatives;
+				if (!(xcomp && clause.getSubject() == i)
+				   && constituent instanceof IndexedConstituent
+				   // the processing of the xcomps is done in Default
+				   // proposition generator.
+				   // Otherwise we get duplicate propositions.
+				   && !clause.getXcomps().contains(i)
+				   && ((i == clause.getVerb() && options.processCcAllVerbs) || (i != clause.getVerb() && options.processCcNonVerbs)))
{
```

```
+                        alternatives = ProcessConjunctions.processCC(depTree, constituent, false, false, Integer.MAX_VALUE, null, null);
+                    } else if (!(xcomp && clause.getSubject() == i)
+                            && clause.getXcomps().contains(i)) {
+                        alternatives = new ArrayList<Constituent>();
+                        DFClausIE xclausIE = new DFClausIE(options);
+                        xclausIE.semanticGraph = semanticGraph;
+                        xclausIE.depTree = depTree;
+                        xclausIE.xcomp = true;
+                        xclausIE.clauses = ((XcompConstituent) clause.getConstituents()
+                                .get(i)).getClauses();
+                        xclausIE.generatePropositions();
+                        for (Proposition p : xclausIE.propositions) {
+                            StringBuilder sb = new StringBuilder();
+                            String sep = "";
+                            for (int j = 0; j < p.constituents.size(); j++) {
+                                if (j == 0)    // to avoid including the subjecct, We
+                                    continue;  // could also generate the prop
+                                // without the subject
+                                sb.append(sep);
+                                sb.append(p.constituents.get(j));
+                                sep = " ";
+                            }
+                            alternatives.add(new TextConstituent(sb.toString(),
+                                    constituent.type));
+                        }
+                    } else {
+                        alternatives = new ArrayList<Constituent>(1);
+                        alternatives.add(constituent);
+                    }
+                    constituents.add(alternatives);
+                }
+
+                // create a list of all combinations of constituents for which a
+                // proposition should be generated
+                includeConstituents.clear();
+                flags.clear();
+                include.clear();
+                for (int i = 0; i < clause.getConstituents().size(); i++) {
+                    Flag flag = clause.getFlag(i, options);
+                    flags.add(flag);
+                    include.add(!flag.equals(Flag.IGNORE));
+                }
+                if (options.nary) {
+                    // we always include all constituents for n-ary ouput
+                    // (optional parts marked later)
+                    includeConstituents.add(include);
+                } else {
+                    // triple mode; determine which parts are required
+                    for (int i = 0; i < clause.getConstituents().size(); i++) {
+                        include.set(i, flags.get(i).equals(Flag.REQUIRED));
+                    }
+
+                    // create combinations of required/optional constituents
+                    new Runnable() {
+                        int noOptional;
+
+                        @Override
+                        public void run() {
+                            noOptional = 0;
+                            for (Flag f : flags) {
+                                if (f.equals(Flag.OPTIONAL))
+                                    noOptional++;
+                            }
+                            run(0, 0, new ArrayList<Boolean>());
+                        }
+
+                        private void run(int pos, int selected, List<Boolean> prefix) {
+                            if (pos >= include.size()) {
+                                if (selected >= Math.min(options.minOptionalArgs,
```

```
+                                                 noOptional)
+                                    && selected <= options.maxOptionalArgs) {
+                                       includeConstituents.add(new ArrayList<Boolean>(
+                                                       prefix));
+                                    }
+                                    return;
+                                }
+                                prefix.add(true);
+                                if (include.get(pos)) {
+                                    run(pos + 1, selected, prefix);
+                                } else {
+                                    if (!flags.get(pos).equals(Flag.IGNORE)) {
+                                        run(pos + 1, selected + 1, prefix);
+                                    }
+                                    prefix.set(prefix.size() - 1, false);
+                                    run(pos + 1, selected, prefix);
+                                }
+                                prefix.remove(prefix.size() - 1);
+                            }
+                        }.run();
+                    }
+
+                    // create a temporary clause for which to generate a proposition
+                    final Clause tempClause = clause.clone();
+
+                    // generate propositions
+                    new Runnable() {
+                        @Override
+                        public void run() {
+                            // select which constituents to include
+                            for (List<Boolean> include : includeConstituents) {
+                                // now select an alternative for each constituent
+                                selectConstituent(0, include);
+                            }
+                        }
+
+                        void selectConstituent(int i, List<Boolean> include) {
+                            if (i < constituents.size()) {
+                                if (include.get(i)) {
+                                    List<Constituent> alternatives = constituents
+                                            .get(i);
+                                    for (int j = 0; j < alternatives.size(); j++) {
+                                        tempClause.getConstituents().set(i,
+                                                        alternatives.get(j));
+                                        selectConstituent(i + 1, include);
+                                    }
+                                } else {
+                                    selectConstituent(i + 1, include);
+                                }
+                            } else {
+                                // everything selected; generate
+                                propositionGenerator.generate(propositions, tempClause,
+                                                include);
+                            }
+                        }
+                    }.run();
+                }
+            }
+
+            public Collection<Proposition> getPropositions() {
+                return propositions;
+            }
+
+            public ArrayList<Map> run() {
+                OutputStream out = System.out;
+                PrintStream dout = new PrintStream(out);
+                ArrayList<Map> al = new ArrayList<Map>();
+                this.detectClauses();
+                try {
```

```
+                this.generatePropositions();
+
+                dout.println(this.getSemanticGraph().toFormattedString()
+                        .replaceAll("\n", "\n#               ").trim());
+
+                dout.print("#   Detected ");
+                dout.print(this.getClauses().size());
+                dout.println(" clause(s).");
+                ObjectMapper mapper = new ObjectMapper();
+                for (Clause clause : this.getClauses()) {
+                    Map svo = clause.svoMap(this.options);
+                    if (svo != null) {
+                      // Object o;
+                      // o = svo.get("TUAid");
+                      // if (o instanceof String) {
+                      //   String TUAid = (String) o;
+                      //   svo.remove("TUAid");
+                      //   svo.put("TUAid", TUAid);
+                      // }
+                      // o = svo.get("metadata");
+                      // if (o instanceof String) {
+                      //   String metadata = (String) o;
+                      //   svo.remove("metadata");
+                      //   svo.put("metadata", metadata);
+                      // }
+                      svo.put("text", this.text);
+                      al.add(svo);
+                      try {
+                        dout.print(mapper.writeValueAsString(svo));
+                      } catch (IOException e) {
+                        e.printStackTrace();
+                      }
+                      dout.println();
+                    }
+                }
+            } catch (java.lang.NullPointerException e) {
+                System.err.println("IGNORING KNOWN PROBLEM: NullPointerException");
+            } catch (java.lang.IndexOutOfBoundsException e) {
+                System.err.println("IGNORING KNOWN PROBLEM: IndexOutOfBoundsException");
+            } catch (java.lang.StackOverflowError e) {
+                System.err.println("IGNORING KNOWN PROBLEM: StackOverflowError");
+            }
+            return al;
+        }
+}
```

Description and code for actor and verb replacements dictionaries

Most of the actor and verb dictionary replacements were accomplished in Python.

The following code converts excel spreadsheets of named-persons into python ‘dict’ files:

```
#!/usr/bin/env python

import pandas as pd
import numpy as np

import json
import argparse


parser = argparse.ArgumentParser(description='Label actors')
parser.add_argument('csv_file', help='CSV file')
parser.add_argument('--label', help='Global label')
```

```
args = parser.parse_args()


df = pd.read_csv(args.csv_file, encoding="latin-1")

# Select everything but the first two columns
df_in = df.ix[:,2:]

# Find places where names are filled in
mask = df_in.isnull().as_matrix()

# Grab place names and roles
locations = df.ix[:, 1]
roles = df.columns

rows, cols = np.where(~mask)
joined_actors = df_in.as_matrix()[rows, cols]

actor_info = []
for (joined_actor, row, col) in zip(joined_actors, rows, cols):
   for actor in [a.strip() for a in joined_actor.split(';')]:
      actor_org_info = [s.strip() for s in actor.split(':')]

      name = actor_org_info[0]
      other_org_name, other_org_role = '', ''

      if len(actor_org_info) == 3:
         other_org_name, other_org_role = actor_org_info[1:]
      elif len(actor_org_info) == 2:
         other_org_role = actor_org_info[1]

      actor_dict = {'name': name,
                'location': locations[rowHigh-Level Orientation Document for CoreNLP Co-reference Resolution and ClausIE SVO Extraction],
                'role': roles[col + 2],
                'class': args.label,
                'other_org': {'name': other_org_name,
                         'title': other_org_role}}
      actor_info.append(actor_dict)


fn = 'output_' + args.csv_file.replace('.csv', '') + '.json'
with open(fn, 'w') as f:
   print('Saving ouput to', fn)
   json.dump(actor_info, f, indent=2)
```

The following code scans through TUA text *by city* and replaces named-actors with “police,” “protester,” or “city.”

```
#!/usr/bin/env python

from jsonlite import jsonlite2json
from replacement_factory import replacement_factory


import json
import sys
import os


DEBUG = False


if len(sys.argv) < 3:
   print("Usage: actor_replacement.py parsed_text.jsonlite actor_dict.json")
```

```
    print()
    print("Actor dict can be generated from CSV using actor_parser.py")
    sys.exit(1)


parsed_jsonl, actor_json = sys.argv[1:]

with open(parsed_jsonl) as f:
    data = jsonlite2json(f)

with open(actor_json) as f:
    repl = json.load(f)


# Set up replacement actor dictionary
repl_dict_named_entities = {entity["name"].lower(): entity["class"].lower() for entity in repl}
repl_dict_named_entities.update({"{} {}".format(entity["role"].lower(),
                                 entity["name"].lower()):
                                 entity["class"].lower() for entity in repl})

multiple_replace_named_entities = replacement_factory(repl_dict_named_entities)

# Here we specify where the dictionary has to be applied
for entry in data:
    for key in entry:
        entry[key] = entry[key].lower()


for i, entry in enumerate(data):
    if 'S' in entry:
        entry['S'] = multiple_replace_named_entities(entry['S'])

    if 'O' in entry:
        entry['O'] = multiple_replace_named_entities(entry['O'])

    if 'tua' in entry:
        entry['tua'] = multiple_replace_named_entities(entry['tua'])

    if (i % 500 == 0):
        print('Progress: %.1f%%\r' % (i / (len(data) - 1) * 100), end='')

print("Progress: 100%")


fn = 'output_' + os.path.split(parsed_jsonl)[-1]
with open(fn, 'w') as f:
    print('Saving ouput to', fn)
    for line in data:
        f.writelines([json.dumps(line), '\n'])
```

The following code scans through *all* Subject and Objects and replaces named-actors with “police,” “protester,” or “city.” It also scans through *all* Verbs and replaces them with a shorter list of verbs.

```
#!/usr/bin/env python

from jsonlite import jsonlite2json

from replacement_factory import replacement_factory

import json
import sys
import re
import os
```

```
DEBUG = False


if len(sys.argv) < 4:
   print("Usage: actor_replacement.py parsed_text.jsonlite words_to_actor.csv words_to_words.csv")
   print()
   print("Actor dict can be generated from CSV using actor_parser.py")
   sys.exit(1)


parsed_jsonl, words_to_actor, words_to_words = sys.argv[1:]

with open(parsed_jsonl) as f:
   data = jsonlite2json(f)


#
# --- Handle words to entity replacements ---
#

import csv
f = csv.reader(open(words_to_actor, encoding="latin-1"))

header = next(f)
replacement_values = header[1:5]
replacements = [line[1:5] for line in f]

# These classes will be parsed out later.  For now, dump them.
replacements = [[item.split(':')[0].strip() for item in line] for line in replacements]

repl_dict_actors = {}
for line in replacements:
   for i, item in enumerate(line):
      if item.strip():
         repl_dict_actors[item.lower()] = replacement_values[i].lower()

#
# --- Handle words to words replacements ---
#

print(words_to_words)
f = csv.reader(open(words_to_words))

header = next(f)
assert('Lemmas' in header[0])

repl_dict_verbs = {}
for line in f:
   words = [w.strip() for w in line[:2]]
   if words[0] and words[1]:
      repl_dict_verbs[words[0].lower()] = words[1].lower()


multiple_replace_actors = replacement_factory(repl_dict_actors)
multiple_replace_verbs = replacement_factory(repl_dict_verbs)


# Here we specify where the dictionary has to be applied
for entry in data:
   for key in entry:
      entry[key] = entry[key].lower()


for i, entry in enumerate(data):
   if DEBUG:
      if any(k in entry['text'] for k in keys):
         print()
         print('<<', entry['text'])
```

```
        print('>>', multiple_replace(entry['text']))

    if 'S' in entry:
        entry['S'] = multiple_replace_actors(entry['S'])

    if 'O' in entry:
        entry['O'] = multiple_replace_actors(entry['O'])

    if 'tua' in entry:
        entry['tua'] = multiple_replace_actors(entry['tua'])

    if 'Lemma' in entry:
        entry['Lemma'] = multiple_replace_verbs(entry['Lemma'])



    if (i % 1000 == 0):
        print('Progress: %.1f%%\r' % (i / (len(data) - 1) * 100), end='')
print("Progress: 100%")


fn = 'output_' + os.path.split(parsed_jsonl)[-1]
with open(fn, 'w') as f:
    print('Saving ouput to', fn)
    for line in data:
        f.writelines([json.dumps(line), '\n'])
```

Data Preparation and Topic Modeling in R – Chapters 4 & 5

The following is the R code used to perform structural topic modeling on protester event text units. It starts by reading in data where each row is an event text unit. Then, data describing the cities in which events occurred are merged with the dataset. Some variables are generated. A date-of-event variable is derived from the article_date and the days (e.g. 'Tuesday') mentioned in the text. Continuous city variables describing Political Opportunity Structures are converted to ordinal variables. Then, Structural Topic Models are estimated and plotted.

```
rm(list=ls())
options(encoding= "UTF-8")
library(jsonlite)
library(plyr)
# #install.packages('reshape')
require(reshape)
#
setwd("~/Documents/Protester_Repertoires")


## Protester.jsonl + Police.jsonl combined to cover ALL cities in current dataset
## removed 4 irrelevant lines with "city" of "Assignments as of 9-21-13" mistakenly included in Police.jsonl
all <- stream_in(file("All.jsonl"))

###GET RID OF NON-TUA ARTEFACTS and focus on one type at a time... and exclude irrelevant cities
all <- subset(all, TUAfile != "NA")
all <- subset(all, TUAtype == 'Protester')


####READ IN TUA TEXT #Thanks, Karthik Ram ###################################
read_file <- function(TUAfile) {
  readChar(TUAfile, file.info(TUAfile)$size)
}
```

```
library(dplyr)
all <- all %>%
 rowwise()  %>%
 mutate(tua = read_file(TUAfile))

#tua is the name of the new column

#THEN, convert pros into a legitimate data.frame #Thanks, Daniel Turek
class(all)
#prosdf <- class(as.data.frame(pros))  #not sure I need that
all <- as.data.frame(all)
class(all)

########################################################################
#########send out for actor dictionary parsing in python

stream_out(all, file("Protester11.jsonl"))

##do the stuff in python

#########read back in with all those replacements in S, O, and tua text, and Lemma
all2<- stream_in(file("final_output_July9_15_reproduction.jsonl"))     #this file is (dumbly) in actor-parser-master-2
##########convert cells with replacements to thier simple actor name
#write a function for this

all2$S[grepl('protesters', all2$S)] <- 'protesters'
all2$O[grepl('protesters', all2$O)] <- 'protesters'
all2$S[grepl('police', all2$S)] <- 'police'
all2$O[grepl('police', all2$O)] <- 'police'
all2$S[grepl('city', all2$S)] <- 'city'
all2$O[grepl('city', all2$O)] <- 'city'

# df$b[grepl('cats', df$b)] <- 'cats'
# df$b[grepl('cats', df$b)] <- 'cats'

#CONCATENATE S, V, O, XCOMP to create S_V_O tokens for later inclusion in topic modeling
all2$svo<-paste(all2$S, all2$Lemma, all2$NEGATED, all2$O, all2$XCOMP, sep = "_")

all2$svocon<-paste(all2$S, all2$Lemma, all2$NEGATED, all2$O, all2$XCOMP, all2$svo, sep = " ")

all2$tua_svo<-paste(all2$tua, all2$svo, sep = " ")

#APPEND svocons to Tua text.
all2$tua_svocon<-paste(all2$tua, all2$S, all2$Lemma, all2$NEGATED, all2$O, all2$XCOMP, all2$svo, sep = " ")

######
#View(all$tua_svo[grepl('Bradley Russell', all$tua_svo)])

###################MERGE on city key#################################
###################MERGE on city key#################################
## knowncities.csv is derived from cityvars2.csv with 4 extraneous
## cities removed (moved into unused extracities.csv) and 5 missing
## cities added (see last 5 entries in knowncities.csv)
known <- read.csv("knowncities.csv", header = TRUE, colClasses = c('character', 'character'))
extra <- read.csv("extracities.csv", header = TRUE, colClasses = c('character', 'character'))

known$key <- tolower(gsub('[^[:alpha:]]', '', paste(known$city,known$state)))

rewrite <- function(x) {
 if (x == "cleveland") return ("clevelandoh")
 if (x == "dayton") return ("daytonoh")
 if (x == "jackson") return ("jacksonms")
 if (x == "allentowncity") return ("allentownpa")
 if (x == "lansingcity") return ("lansingmi")
 if (x == "pensacolaflorida") return ("pensacolafl")
 if (x == "cedarfallscedarvalley") return ("cedarfallsia")
 if (x == "everettcity") return ("everettwa")
 if (x == "coachella") return ("coachellavalleyca")
 if (x == "lexingtonfayetteky") return ("lexingtonky")
```

```
 if (x == "louisville") return ("louisvillejeffersoncountyky")
 if (x == "newyork") return ("newyorkcityny")
 if (x == "nashville") return ("nashvilledavidsontn")
 string = x
 inlist = known$key
 abbr <- substr(string, 1, nchar(string)-2)
 found <- inlist[grepl(paste("^", string, sep=""), inlist)]
 if (length(found) == 0) {
  found <- inlist[grepl(paste("^", abbr, "$", sep=""), inlist)]
 }
 if (length(found) == 1) {
  return(found)
 } else if (length(found > 1)) {
  warning(paste("found duplicates: ", string))
  return(found)
 } else {
  warning(paste("did not find: ", string))
  return(F)
 }
}

all2$key <- mapply(rewrite, tolower(gsub('[^[:alpha:]]', '', all2$city)))

cityvars3 <- read.csv("cityvars3.csv", header = TRUE, colClasses = c('character', 'character', 'character', 'character', 'integer', 'character',
'integer', 'numeric', 'numeric', 'numeric', 'character', 'character', 'character', 'numeric', 'numeric', 'numeric', 'numeric', 'numeric',
'numeric', 'factor', 'numeric'), fileEncoding = "UTF-8")
all3 <- merge(all2, cityvars3, by=c("key"), all.x=TRUE)
unique(all3$key) ## return all 185 cities
unique(paste(all3$Clean_city,all3$State)) ## returns all 185 cities... (because uniqueness MUST include state, not just city)
unique(all3$Clean_city) ## as expected returns ONLY 179 cities... because some states have same city name!!!!


#get rid of cities outside our data set
all3 <- subset(all3, key!="lakeworthfl")
all3 <- subset(all3, key!="athensga")
all3 <- subset(all3, key!="lancasterpa")
all3 <- subset(all3, key!="clevelandtn")


##################################################
##################################################


######################
#DATES STUFF
######################

#First, Cleaning
# Replace "None" with Jan 1, 1999 so it is a date by which you can filter later.
class(all3$article_date)
all3$article_date<-as.character(all3$article_date)

Broken <- subset(all3, article_date=="none")
Broken <- subset(all3, article_date=="undated")
Broken <- subset(all3, article_date=="")
Broken <- subset(all3, (nchar(article_date) > 8))
Broken <- subset(all3, article_date=="???")
Broken <- subset(all3, Next_Election==NA)
rm(Broken)

all3clean <- subset(all3, article_date!="")
all3clean <- subset(all3clean, article_date!="+*")
all3clean <- subset(all3clean, article_date!="???")
all3clean <- subset(all3clean, article_date!="none")
all3clean <- subset(all3clean, article_date!="undated")
all3clean <- subset(all3clean, (nchar(article_date) < 9))
all3clean <- subset(all3clean, article_date!="12-12-")  #or was it 12-2
```

```
View(unique(all3clean$article_date))

class(all3clean$article_date)
#as.character(all3clean$article_date)

####################################
#Some FUNCTIONS for DATE EXTRACTION
####################################
#This function takes a formatted date as Y-m-d and returns weekday
day_from_date <- function(adate = NULL) {
 if (!is.null(adate)) {
   day_list <- c("Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday",
                 "Saturday")
   dow <- try(which(grepl(weekdays(as.Date(adate, "%m-%d-%y")), day_list)), silent=TRUE)
   if(inherits(dow, "try-error")) dow <- NA
   return(dow)
 }
}

# This function takes some text and returns the position on the weekday list
# So return_days("Monday Tuesday Friday")
# will return 2 (for Monday)
return_days <- function(text) {
 if(!is.na(text)){
 if(is.character(text)) {
   day_list <- c("Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday",
                 "Saturday")
   res <- as.numeric(which(sapply(day_list, grepl, text, ignore.case = TRUE)))
   if (length(res) == 0) {
     res <- NA
   } else {
     res<-res[1] # we can add more handling here to deal with multiple weekday mentions
   }
 }
 } else res <- NA
 return(res)
}

# Function takes weekday difference between article_date and text_date and
# calculates approximate date.
day_in_text <- function(article_day, aday, pdate) {
 if(!is.na(pdate)){
   date_diff <- aday - pdate
   if (date_diff > 0) {
     out <- as.Date(article_day, "%m-%d-%y") - date_diff
   } else {
     dd <- ifelse(date_diff != 0, date_diff + 7, 0)
     out <- as.Date(article_day, "%m-%d-%y") - dd
   }
 } else{
   out <- try(as.Date(article_day, "%m-%d-%y") - 1, silent=TRUE)
   if(inherits(out, "try-error")) out <- NA
 }
 return(out)
}

########clean out weird dates##########
View(unique(all3clean$article_date))
#######################################
all3clean <- subset(all3clean, article_date!="3-30-12")
all3clean <- subset(all3clean, article_date!="10-22-13")
all3clean <- subset(all3clean, article_date!="10-28-13")
all3clean <- subset(all3clean, article_date!="7-17-12")
all3clean <- subset(all3clean, article_date!="9-24-12")
all3clean <- subset(all3clean, article_date!="2-27-12")
all3clean <- subset(all3clean, article_date!="11-1-13")
all3clean <- subset(all3clean, article_date!="10-26-13")
all3clean <- subset(all3clean, article_date!="11-17-12")
all3clean <- subset(all3clean, article_date!="11-7-10")
```

```
all3clean <- subset(all3clean, article_date!="12-13-1")
all3clean <- subset(all3clean, article_date!="12-11-12")



##############################################
#NOW, to extract DATES from tua text daynames
##############################################
library(lubridate)
#library(dplyr)
library(readr)

all3clean$article_date<-as.character(all3clean$article_date)

published_day <-  sapply(all3clean$article_date, day_from_date)

text_day <- sapply(all3clean$tua_svo, return_days)
guessing_date_in_text <- mapply(function(a,b,c)
  day_in_text(article_day=a, aday=b, pdate=c),
  a=all3clean$article_date,
  b=published_day,
  c=text_day,
  SIMPLIFY=FALSE)

guessing_date_in_text <- lapply(guessing_date_in_text, function(x) as.character(x))
guessing_date_in_text <- c(do.call(rbind, guessing_date_in_text))
all3clean$text_date <- guessing_date_in_text
all3clean$text_date <- as.Date(all3clean$text_date, "%Y-%m-%d")
View(unique(all3clean$text_date))



##############################
#DAYS running variables
##############################
#days camped
#text_date - campstartdate
class(all3clean$Campaign.Start.Date)
#character
class(all3clean$text_date)
#Date
all3clean$camp_date<-as.Date(all3clean$Campaign.Start.Date, "%m/%d/%y")
all3clean$dayscamped<- all3clean$text_date - all3clean$camp_date
all3clean$dayscamped<-as.numeric(all3clean$dayscamped)
View(unique(all3clean$dayscamped))
View(unique(all3clean$camp_date))
View(unique(all3clean$Campaign.Start.Date))
View(unique(all3clean$text_date))



#days since zucotti
all3clean$dayszuc<-all3clean$text_date - as.Date("2011-09-17")
all3clean$dayszuc<-as.numeric(all3clean$dayszuc)
View(unique(all3clean$dayszuc))

##############################################
#ReFactor POS Variables
##############################################



#make so Gov't_Type captures increasing openness from Mayoral to Commission + state capital
all3clean$Govt_Type<-factor(all3clean$Govt_Type, levels=c("MC", "CM", "CO")) #if this doesn't work, check that cityvars2 has been
updated
#convert levels to numbers
all3clean$govt3<-as.numeric(all3clean$Govt_Type)
#add the state capital number
```

```
all3clean$pos1<- all3clean$govt3 + all3clean$State.Capital
View(unique(all3clean$govt3))
View(unique(all3clean$State.Capital))
View(unique(all3clean$pos1))


######
#Electoral instability
#######
#camp start date - prior election. wihtin 3 months... 3, within 6... 2, within 12 months... 1, otherwise 0
#upcoming election - camp start date ... same as above
#then add them together
#class(all3clean$Next_Election)
#View(unique(all3clean$Next_Election))
#add an arbitrary daydate "01" to all the Next_Election data
all3clean$Next_Election<-as.Date(paste(all3clean$Next_Election,"-01", sep=""))
View(unique(all3clean$Next_Election))
table(all3clean$Next_Election)
#set as Date type and calculate weeks till next election
all3clean$Next_Election<-as.Date(all3clean$Next_Election, "%Y-%m-/%d/")
all3clean$weekstill<-as.numeric(difftime(all3clean$Next_Election, as.Date("2011-09-17"), units = "weeks"))
View(unique(all3clean$weekstill))

#calculates weeks since last election
all3clean$Prior_Election<-as.Date(all3clean$Prior_Election, "%m/%d/%y")
class(all3clean$Prior_Election)
all3clean$weekssince<-as.numeric(difftime(as.Date("2011-09-17"), all3clean$Prior_Election, units = "weeks"))
View(unique(all3clean$weekssince))



####
library(Hmisc)
all3clean$stability3<-(all3clean$weekssince + all3clean$weekstill)
all3clean$stability3<- as.numeric(cut2(all3clean$stability3, g=3))
table(all3clean$stability3)



#### code below is what I used in the first run ###### correcting as writing Chapter 6
# a <- 20 - floor(all3clean$weekstill / 13)
# all3clean$weekstill<-floor(a/5)
#
# b <- 20 - floor(all3clean$weekssince / 13)
# all3clean$weekssince<-floor(b/5)
#
# all3clean$stability<-(all3clean$weekssince + all3clean$weekstill)*2/3
# View(unique(all3clean$stability))
#
# all3clean$stability<-(all3clean$weekssince + all3clean$weekstill)
# table(all3clean$stability)
# # make stability easier to work with
# class(all3clean$stability)
# all3clean$stability2<-ceiling(all3clean$stability *3/4-1)
# table(all3clean$stability2)




#######
#Obama voters as pool of supporters
#######
all3clean$pool<-all3clean$Population*all3clean$Obama_Vote2008/100

#cut it up into equal sized bins
install.packages('Hmisc')
library(Hmisc)
```

```
all3clean$pool2 <- as.numeric(cut2(all3clean$pool, g=5))
View(unique(all3clean$pool2))
all3clean$pool3 <- as.numeric(cut2(all3clean$pool, g=3))

all3clean$pop3 <- as.numeric(cut2(all3clean$Population, g=3))
all3clean$Obama3 <- as.numeric(cut2(all3clean$Obama_Vote2008, g=3))
View(unique(all3clean$pop))
View(unique(all3clean$Obama))

class(all3clean$Obama)

class(all3clean$pop)


save(all3clean,file="Diss4.Rda")
load("Diss4.Rda") ########################################################


save(all3clean, file="Diss6.Rda")  #this is a reproduction with 3-value variables and using the later iteration of the Stefan's dictionary
#####################################################
#####################################################
#convert from SVO data.frame to TUA data.frame
#####################################################
#####################################################
#####################################################
View(unique(all3clean$tua))

processNicksDF <- function(df) {
  out <- data.frame(tua = character(),
                    text = character(),
                    key = character(),
                    pos1 = numeric(),
                    stability3 = numeric(),
                    Obama3 = numeric(),
                    dayscamped = numeric(),
                    dayszuc = numeric(),
                    pop3 = numeric(),
                    Obama_Vote2008 = numeric(),
                    Population = numeric(),
                    pool3 = numeric(),
                    stringsAsFactors = FALSE)
  for(e in unique(df$tua)) {
    tempDF <- df[df$tua==e,]
    prep <- data.frame(tua = e,
                       text = paste(e, paste(tempDF$svocon, collapse=' ')),
                       key = tempDF$key[1],
                       pos1 = tempDF$pos1[1],
                       stability3 = tempDF$stability3[1],
                       Obama3 = tempDF$Obama3[1],
                       dayscamped = tempDF$dayscamped[1],
                       dayszuc = tempDF$dayszuc[1],
                       pop3 = tempDF$pop[1],
                       Obama_Vote2008 = tempDF$Obama_Vote2008[1],
                       Population = tempDF$Population[1],
                       pool3 = tempDF$pool3[1],
                       stringsAsFactors = FALSE)
    out <- rbind(out, prep)
  }
  return(out)
}

#build TUA-based dataframe
df <- processNicksDF(all3clean)
#
#
#               SAVE!!!!
#####################################################################################################
###########
```

```
##########################################################################################################
##########
# save(df,file="TUAdf3.Rda")
# load("TUAdf3.Rda")
# save(df,file="TUAdf4.Rda")  #includes city 'key' variable now!
save(df,file="TUAdf5.Rda")   #fixes while writing Chapter 6
##########################################################################################################
##########

View(df$text)
#clean up svo stuff
df$text <-gsub('NA', '', df$text)
df$text<-gsub('_+', '_', df$text)
df$text<-gsub("'", '', df$text)
df$text<-gsub('-rsb-', '', df$text)
df$text<-gsub('-lsb-', '', df$text)
df$text<-gsub('-lrb-', '', df$text)
df$text<-gsub('-rrb-', '', df$text)
df$text<-gsub(',', '', df$text)
df$text<-gsub(':', '', df$text)
df$text<-gsub('-rsb-', '', df$text)

#fix some mistakes from prior substitutions
df$text<-gsub('protestersers', 'protesters', df$text)
df$text<-gsub('protestersed', 'protested', df$text)
df$text<-gsub('protestersing', 'protesting', df$text)
df$text<-gsub('protesterors', 'protesters', df$text)
df$text<-gsub('protestors', 'protesters', df$text)
df$text<-gsub('police police police', 'police police', df$text)
df$text<-gsub('protesters protesters protesters', 'protesters protesters', df$text)
df$text<-gsub('police police police', 'police police', df$text)
df$text<-gsub('protesters protesters protesters', 'protesters protesters', df$text)
df$text<-gsub('police police police', 'police police', df$text)
df$text<-gsub('protesters protesters protesters', 'protesters protesters', df$text)
df$text<-gsub('police police', 'police', df$text)
df$text<-gsub('protesters protesters', 'protesters', df$text)


#reduces some special classes of objects like citynames, days, parks, plazas, and some other named entitites
df$text<-gsub("bo didley community plaza|frank ogawa plaza|calder plaza|cesar chavez plaza|hemming plaza|san jacinto plaza", 'xplaza',
df$text)
df$text<-gsub("Albany|Albuquerque|Allentown|philly|Anaheim|Anchorage|Ann
Arbor|Arcata|Asheville|Ashtabula|Atlanta|Augusta|Aurora|Austin|Bakersfield|Baltimore|Bangor|Baton
Rouge|Bellingham|Bend|Bethlehem|Binghamton|Birmingham|Bloomington|Boise|Boston|Bowling
Green|Brattleboro|Buffalo|Burlington|Canton|Cedar Falls|Cedar Rapids|Chapel
Hill|Carrboro|Charleston|Charleston|Charlotte|Chattanooga|Chicago|Chico|Cincinnati|Claremont|Clarksville|Cleveland
        |Coachella Valley|Colombus|Colorado Springs|Columbia|Corpus Christi|Dade City|Dallas|Dayton|Daytona Beach|Denver|Des
Moines|Detroit|Dover|Duluth|Easton|El Paso|Eugene|Eureka|Everett|Fairbanks|Fayetteville|Flint|Fort Collins|Fort Lauderdale|Fort
Myers|Fort Wayne|Fort Worth|Frederick|Fresno|Gainesville|Grand
Rapids|Greeley|Greensboro|Harrisburg|Hartford|Honolulu|Houston|Huntington|Indianapolis|Iowa City|Irvine|Ithaca|Jackson|
        Jacksonville|Jersey City|Johnson City|Kalamazoo|Kansas City|Lansing|Las Cruces|Las Vegas|Lawrence|Lexington|Lincoln|Little
Rock|Long Beach|Los Angeles|Louisville|Jefferson
County|Lubbock|Madison|Memphis|Merced|Miami|Milwaukee|Minneapolis|Missoula|Mobile|Mosier|Muncie|Murfreesboro|Muskegon|N
ashville|Davidson|New Haven|New Orleans|New Paltz|New York City|Newark|Norfolk|Norman|Northampton|Oakland|Ogden|Oklahoma
City|Olympia|
        Omaha|Orlando|Palo
Alto|Pensacola|Petaluma|Philadelphia|Phoenix|Pittsburgh|Pocatello|Portland|Portland|Poughkeepsie|Providence|Raleigh|Richmond|Ri
verside|Rochester|Sacramento|Salem|San Antonio|San Diego|San Francisco|San Jose|San Leandro|San Ramon|Santa Ana|Santa
Cruz|Santa Fe|Santa Rosa|Scranton|Seattle|Sebastopol|Sonoma|South Bend|St.
Louis|Stockton|Syracuse|Tacoma|Tallahassee|Tampa|Toledo|Trenton|Tucson|Tulsa|Utica|
        Virginia Beach|Walnut Creek|Washington|West Palm
Beach|Wichita|Wilmington|Wilmington|Worcester|Youngstown|Yuma|Venice", "cityname", df$text, ignore.case=TRUE)
df$text<-gsub("California", 'xstate', df$text)
df$text<-gsub("cityname movement|cityname protesters|cityname demonstrators|cityname activists|occupy cityname", 'protesters',
df$text)
df$text<-gsub("cityname deputies|cityname police department|cityname police", 'police', df$text)
df$text<-gsub("bank of america| indep of america|wells fargo|chase bank", 'xbank', df$text)

df$text<-gsub("brooklyn bridge|cityname street bridge", 'xbridge', df$text)
```

```
df$text<-gsub("jessica reznieck", 'xprotester', df$text)
df$text<-gsub("saturday|sunday", 'xweekendday', df$text)
df$text<-gsub("monday|tuesday|wednesday|thursday|friday", 'xweekday', df$text)
df$text<-gsub("grand circus park|academy park|lafayette park|bubier park| thomas square park|smith park|zucotti park|cityname-
jefferson park|woodruff park|oak cliff park|pioneer park", 'xpark', df$text)
df$text<-gsub("cielo vista mall|valley malls", 'xshop', df$text)
View(df$text)


#remove all rows with missing data
df <- na.omit(df)

#save(df,file="DissTUAstm_Sunday.Rda")
save(df,file="DissTUAstm_7_9_15.Rda") # fixing plots while writing Chapter 6



rm(all)
rm(all3clean)




##################################################
#STRUCTURAL TOPIC MODELING
##################################################
#load("DissTUAstm2.Rda")
load("DissTUAstm_Wednesday.Rda")
load("DissTUAstm_Sunday.Rda")

#install.packages('stm')
library(stm)
install.packages('igraph')
library(igraph)
remove.packages('igraph')

################################################
#PREPROCESSING
################################################

#stemming/stopword removal, etc. also duplicates all data for a metadata file created in next step
?textProcessor
processed <- textProcessor(df$text, metadata=df, stem=FALSE, removepunctuation=FALSE)

#structure and index for usage in the stm model. Verify no-missingness. can remove low frequency words using 'lower.thresh' option.
See ?prepDocuments for more info
?prepDocuments
out <- prepDocuments(processed$documents, processed$vocab, processed$meta, lower.thresh = 0)


#IF YOU WANT TO FIDDLE WITH LOWER THRESHHOLDS, take a look at how many words and documents would be removed with
different lower.thresholds    !!! check Error: could not find function "plotRemoved"
#plotRemoved(processed$documents, lower.thresh=seq(1,200, by=100))

#CREATES docs and (identical) meta dataframes and a vocab vector of all words in the corpus -- these are the elements of stm
docs <- out$documents
vocab <- out$vocab
meta <-out$meta

################################################
#RUN AND CHOOSE THE BEST TOPIC MODEL
################################################

#to test the affects of POS on operational/tactical activity, we use indicators of the various aspects of POS
#number of power centers is represented by Gov't Type
#openness to new actors is represented by Months
#instability of current political alignments is also represented by Mayor Strength
#availability of influential allies or supporters (will be added in after humans handcode for such allies)
#the extent to which the regime represses should not be folded into POS, but is open for observation and analysis
```

```
#ADD in POS variables and date


#let STM help you compare a number of models side by side. It will keep the models that don't stink (i.e. that converge quickly)

# lifecourse40Select <- selectModel(out$documents,out$vocab,K=40,prevalence =~ pos1 + pool2 + stability2 + s(dayscamped),
max.em.its=250, data=out$meta, runs=20)
#
#

plotModels(lifecourse40Select)

forty4<-lifecourse40Select$runout[[4]]
forty1<-lifecourse40Select$runout[[1]]
forty3<-lifecourse40Select$runout[[3]]
forty2<-lifecourse40Select$runout[[2]]

save(lifecourse40, file="stmlifecourse.Rda")



##############################################
#BEGIN INTERPRETING THE MODELS
##############################################

labelTopics(lifecourse40, topics=c(1, 4, 5, 7, 10, 12, 19, 22, 24, 27, 29, 35, 38, 40), n=10)

forty4$settings$seed
#4813083

###WORDCLOUD for a specified TOPIC
cloud(forty4, topic=9)
cloud(it3, topic=9, max.words=50)
cloud(lifecourse40, topic=40)

cloud(lifecourse40, topic=19)

?cloud
###Read DOCUMENTS that are highly correlated with the topics you specify using findThoughts() function
#object 'thoughts1' contains 2 documents about topic 1. 'texts=shortdoc,' gives you just the first 250 words
thoughts1<-findThoughts(it, texts=df$tua, n=1, topics=13)$docs[[1]]
#will show you the output
?findThoughts
plotQuote(thoughts1, width=40, main="Topic 9")
?plotQuote
#how about more documents for more of these topics?
thoughts7 <- findThoughts(it, texts=shortdoc,n=2, topics=7)$docs[[1]]
thoughts10 <- findThoughts(it, texts=shortdoc,n=2, topics=10)$docs[[1]]
thoughts4 <- findThoughts(it, texts=shortdoc,n=2, topics=4)$docs[[1]]
#And in a 2X2 table? We like 2X2 tables!  --- Note: this command will force all remaining plots into a 2X2 table format
par(mfrow = c(2, 2),mar=c(.5,.5,1,.5))
plotQuote(thoughts1, width=40, main="Topic 1")
plotQuote(thoughts4, width=40, main="Topic 4")
plotQuote(thoughts7, width=40, main="Topic 7")

##see PROPORTION OF EACH TOPIC in the entire CORPUS. Just insert your STM output
plot.STM(lifecourse40, type="summary", main = " ", topics= c(1:40), xlim=c(0,.4))
plot.STM(lifecourse40, type="hist", topics= c(1:20), xlim=c(0,.4))
?plot.STM
##see GRAPHICAL NETWORK DISPLAY of how closely related topics are to one another, (i.e., how likely they are to appear in the same
document) Requires 'igraph' package
install.packages('huge')
library(huge)
mod.out.corr<-topicCorr(lifecourse40, method = c("simple"))
plot.topicCorr(mod.out.corr)
?topicCorr
```

```
#############################################################################################################
########################
# STM: SEE HOW PREVALENCE OF TOPICS VARIES ACROSS DOCUMENTS ACCORDING TO DOCUMENT COVARIATES (METADATA)
#############################################################################################################
########################
preplc40 <- estimateEffect(1:40 ~ pos1 + pool2 + stability2 + s(dayscamped), lifecourse40, meta=meta, uncertainty = "Global")

cols <-brewer.pal(4, "Set1")[c(3,4,2,1)]
plot.estimateEffect(preplc40, covariate = "dayscamped", topics = c(38, 5 ,19),
              model=lifecourse40, method="continuous",
              xlab="Days since local camp established",
              xlim=c(-60,166), labeltype = "custom",
              linecol=cols, printlegend=FALSE)

legend("topleft", c('Weekend Gatherings', 'Weekday Marching',
              'Encampment Activities'),
            lwd=2, col=cols, bty="n")



cols <-brewer.pal(4, "Set1")[c(2,1,3,4)]
plot.estimateEffect(preplc40, covariate = "dayscamped", topics = c(12, 1, 35),
              model=lifecourse40, method="continuous",
              xlab="Days since local camp established",
              xlim=c(-60,166), labeltype = "custom",
              linecol=cols, printlegend=FALSE)

legend("topleft", c('Rallies', 'Demonstrations', 'Labor Alliances'),
     lwd=2, col=cols, bty="n")


cols <-brewer.pal(4, "Set1")[c(2,3,1,4)]
plot.estimateEffect(preplc40, covariate = "dayscamped", topics = c(27, 24),
              model=lifecourse40, method="continuous",
              xlab="Days since local camp established",
              xlim=c(-60,166), labeltype = "custom", printlegend=FALSE)

legend("topleft", c('City Hall Targeting', 'Bank Targeting'),
     lwd=2, bty="n")



cols <-brewer.pal(4, "Set1")[c(2,1,3,4)]
plot.estimateEffect(preplc40, covariate = "dayscamped", topics = c(4, 29, 22),
              model=lifecourse40, method="continuous",
              xlab="Days since local camp established",
              xlim=c(-60,166), labeltype = "custom",
              linecol=cols, printlegend=FALSE)
              printlegend=FALSE)

legend("topleft", c('Sidewalk Contestations', 'Curfew Disputes', 'Traffic Battles'),
     lwd=2, col=cols, bty="n")



cols <-brewer.pal(4, "Set1")[c(2,1,3,4)]
plot.estimateEffect(preplc40, covariate = "dayscamped", topics = c(7, 40),
              model=lifecourse40, method="continuous",
              xlab="Days since local camp established",
              xlim=c(-60,166), labeltype = "custom",
              linecol=cols, printlegend=FALSE)

legend("topleft", c('Arrests', 'Standoffs with Riot Gear'),
     lwd=2, col=cols, bty="n")
```

```
cols <-brewer.pal(5, "Set1")[c(3,4,5,2,1)]
plot.estimateEffect(preplc40, covariate = "dayscamped", topics = c(4, 29, 22, 7, 40),
                model=lifecourse40, method="continuous",
                xlab="Days since local camp established",
                xlim=c(-60,166), labeltype = "custom",
                linecol=cols,
                lwd=4,
#                 custom.labels = c('Sidewalk Contestations', 'Curfew Disputes','Arrests', 'Traffic Battles', Standoffs with Riot Gear'))
printlegend=FALSE)

legend("topleft", c('Sidewalk Contestations', 'Curfew Disputes','Street Battles', 'Arrests', 'Standoffs with Riot Gear'),
    lwd=2, col=cols, bty="n")


# plot.estimateEffect(preplc40, covariate = "dayscamped", topics = c(30, 43 ,8 ),
#                 model=it, method="continuous",
#                 xlab="Days sicne local camp established",
#                 main="Activity prevalence since local camps tart",
#                 xlim=c(-60,166), labeltype = "custom",
#                 custom.labels = c('target city hall', 'riot cops',
#                                   'arrests'))
#
# plot.estimateEffect(preplc40, covariate = "dayscamped", topics = c(30, 43 ,8 ),
#                 model=it, method="continuous",
#                 xlab="Days sicne local camp established",
#                 main="Activity prevalence since local camps tart",
#                 xlim=c(-60,166), labeltype = "custom",
#                 custom.labels = c('target city hall', 'riot cops',
#                                   'arrests'))
#
# plot.estimateEffect(preplc40, covariate = "dayscamped", topics = c(30, 43 ,8 ),
#                 model=it, method="continuous",
#                 xlab="Days sicne local camp established",
#                 main="Activity prevalence since local camps tart",
#                 xlim=c(-60,166), labeltype = "custom",
#                 custom.labels = c('target city hall', 'riot cops',
#                                   'arrests'))



########################
########################
########################
#Chapter 5
########################
########################
########################
preplc40 <- estimateEffect(1:40 ~ pos1 + pool2 + stability2 + s(dayscamped), lifecourse40, meta=meta, uncertainty = "Global")

#Simple plots of topic prevalence over time at different levels of covariate. These are NOT time*covariate interactions. Notice how they
stack

f <- function(){
 layout(matrix(1:3, ncol=1, byrow=TRUE))
 par(oma=c(4,6,2,1), mar=c(1,1,1,1), mgp=c(2,1,0),
    cex.axis=1.5)
#Weekend Rallies by size of Obama supporting population
plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(38),
                method="continuous", moderator="pool2", moderator.value=c(1), add=FALSE, ylim=c(-.05,.2), ylab="Expected Topic
Proportion", xlab="Days since local camp established",
                xlim=c(-60,166), xaxt="n", linecol='red', printlegend=FALSE)
plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(38),
                method="continuous", moderator="pool2", moderator.value=c(3), add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(38),
```

```
        method="continuous", moderator="pool2", moderator.value=c(5), add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
legend("topleft", c('Smaller, Less Liberal Population', 'Modal Population', 'Large, Liberal Population'),
    lwd=2, col=c('red', 'blue', 'darkgreen'), bty="n", cex=1)


#Camp Activity by size of Obama supporting population
plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(19),
        method="continuous", moderator="pool2", moderator.value=c(1), add=FALSE, ylim=c(-.2,.25), ylab="Expected Topic
Proportion", xlab="Days since local camp established",
        xlim=c(-60,166), xaxt="n", linecol='red', printlegend=FALSE)
plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(19),
        method="continuous", moderator="pool2", moderator.value=c(3), add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(19),
        method="continuous", moderator="pool2", moderator.value=c(5), add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
mtext("Expected Topic Proportion", 2, line=4)


#Weekday Marches by size of Obama supporting population
plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(5),
        method="continuous", moderator="pool2", moderator.value=c(1), add=FALSE, ylim=c(-.05,.2), ylab="Expected Topic
Proportion", xlab="Days since local camp established",
        xlim=c(-60,166), linecol='red', printlegend=FALSE)
plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(5),
        method="continuous", moderator="pool2", moderator.value=c(3), add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(5),
        method="continuous", moderator="pool2", moderator.value=c(5), add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
mtext("Days Since Local Camp Established", 1, line=3)
}

pdf.f(f, file= '/Users/nickbadams/Dropbox/NickDiss/Chapter 4 Figures and Tables/pop_activity_w3.pdf',
    width=4, height=7)



g <- function(){
 layout(matrix(1:3, ncol=1, byrow=TRUE))
 par(oma=c(4,6,2,1), mar=c(1,1,1,1), mgp=c(2,1,0),
    cex.axis=1.5)
#Rallies by size of Obama supporting population
plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(12),
        method="continuous", moderator="pool2", moderator.value=c(1), add=FALSE, ylim=c(-.1,.15), ylab="Expected Topic
Proportion", xaxt="n",
        xlim=c(-60,166), linecol='red', printlegend=FALSE)
plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(12),
        method="continuous", moderator="pool2", moderator.value=c(3), add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(12),
        method="continuous", moderator="pool2", moderator.value=c(5), add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
legend("topleft", c('Smaller, Less Liberal Population', 'Modal Population', 'Large, Liberal Population'),
    lwd=2, col=c('red', 'blue', 'darkgreen'), bty="n", cex=1)

#Demonstrations by size of Obama supporting population
plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(1),
        method="continuous", moderator="pool2", moderator.value=c(1), add=FALSE, ylim=c(-.1,.125), ylab="Expected Topic
Proportion",  xaxt="n",
        xlim=c(-60,166), linecol='red', printlegend=FALSE)
plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(1),
        method="continuous", moderator="pool2", moderator.value=c(3), add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(1),
        method="continuous", moderator="pool2", moderator.value=c(5), add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
mtext("Expected Topic Proportion", 2, line=4)
```

```
#Labor Alliances by size of Obama supporting population
plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(35),
              method="continuous", moderator="pool2", moderator.value=c(1), add=FALSE, ylim=c(-.12,.125), ylab="Expected Topic
Proportion", xlab="Days since local camp established",
              xlim=c(-60,166), linecol='red', printlegend=FALSE)
plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(35),
              method="continuous", moderator="pool2", moderator.value=c(3), add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(35),
              method="continuous", moderator="pool2", moderator.value=c(5), add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
mtext("Days Since Local Camp Established", 1, line=3)

}

pdf.f(g, file= '/Users/nickbadams/Dropbox/NickDiss/Chapter 4 Figures and Tables/pop_rallies_w3.pdf',
    width=4, height=7)


h <- function(){
 layout(matrix(1:3, ncol=1, byrow=TRUE))
 par(oma=c(4,6,2,1), mar=c(1,1,1,1), mgp=c(2,1,0),
   cex.axis=1.5)
 #Rallies by size of Obama supporting population
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(27),
              method="continuous", moderator="pool2", moderator.value=c(1), add=FALSE, ylim=c(-.1,.15), ylab="Expected Topic
Proportion", xaxt="n",
              xlim=c(-60,166), linecol='red', printlegend=FALSE)
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(27),
              method="continuous", moderator="pool2", moderator.value=c(3), add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(27),
              method="continuous", moderator="pool2", moderator.value=c(5), add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 legend("topleft", c('Smaller, Less Liberal Population', 'Modal Population', 'Large, Liberal Population'),
     lwd=2, col=c('red', 'blue', 'darkgreen'), bty="n", cex=1)

 #Demonstrations by size of Obama supporting population
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(24),
              method="continuous", moderator="pool2", moderator.value=c(1), add=FALSE, ylim=c(-.1,.125), ylab="Expected Topic
Proportion",  xaxt="n",
              xlim=c(-60,166), linecol='red', printlegend=FALSE)
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(24),
              method="continuous", moderator="pool2", moderator.value=c(3), add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(24),
              method="continuous", moderator="pool2", moderator.value=c(5), add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 mtext("Expected Topic Proportion", 2, line=4)


 #Labor Alliances by size of Obama supporting population
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(4),
              method="continuous", moderator="pool2", moderator.value=c(1), add=FALSE, ylim=c(-.12,.125), ylab="Expected Topic
Proportion", xlab="Days since local camp established",
              xlim=c(-60,166), linecol='red', printlegend=FALSE)
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(4),
              method="continuous", moderator="pool2", moderator.value=c(3), add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(4),
              method="continuous", moderator="pool2", moderator.value=c(5), add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 mtext("Days Since Local Camp Established", 1, line=3)

}
pdf.f(h, file= '/Users/nickbadams/Dropbox/NickDiss/Chapter 4 Figures and Tables/pop_targeting_w3.pdf',
    width=4, height=7)
```

```
i <- function(){
 layout(matrix(1:3, ncol=1, byrow=TRUE))
 par(oma=c(4,6,2,1), mar=c(1,1,1,1), mgp=c(2,1,0),
   cex.axis=1.5)
 #Rallies by size of Obama supporting population
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(22),
            method="continuous", moderator="pool2", moderator.value=c(1), add=FALSE, ylim=c(-.1,.15), ylab="Expected Topic
Proportion", xaxt="n",
            xlim=c(-60,166), linecol='red', printlegend=FALSE)
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(22),
            method="continuous", moderator="pool2", moderator.value=c(3), add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(22),
            method="continuous", moderator="pool2", moderator.value=c(5), add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 legend("topleft", c('Smaller, Less Liberal Population', 'Modal Population', 'Large, Liberal Population'),
     lwd=2, col=c('red', 'blue', 'darkgreen'), bty="n", cex=1)

 #Demonstrations by size of Obama supporting population
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(29),
            method="continuous", moderator="pool2", moderator.value=c(1), add=FALSE, ylim=c(-.1,.125), ylab="Expected Topic
Proportion",  xaxt="n",
            xlim=c(-60,166), linecol='red', printlegend=FALSE)
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(29),
            method="continuous", moderator="pool2", moderator.value=c(3), add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(29),
            method="continuous", moderator="pool2", moderator.value=c(5), add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 mtext("Expected Topic Proportion", 2, line=4)


 #Labor Alliances by size of Obama supporting population
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(7),
            method="continuous", moderator="pool2", moderator.value=c(1), add=FALSE, ylim=c(-.12,.125), ylab="Expected Topic
Proportion", xlab="Days since local camp established",
            xlim=c(-60,166), linecol='red', printlegend=FALSE)
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(7),
            method="continuous", moderator="pool2", moderator.value=c(3), add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(7),
            method="continuous", moderator="pool2", moderator.value=c(5), add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 mtext("Days Since Local Camp Established", 1, line=3)

}
pdf.f(i, file= '/Users/nickbadams/Dropbox/NickDiss/Chapter 4 Figures and Tables/pop_dusrupt_w3.pdf',
    width=4, height=7)



#########################################################
#########################################################
#Activity by pos1
#########################################################
#########################################################

j <- function(){
 layout(matrix(1:3, ncol=1, byrow=TRUE))
 par(oma=c(4,6,2,1), mar=c(1,1,1,1), mgp=c(2,1,0),
   cex.axis=1.5)
 #Weekend Rallies by size of # independent power centers
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(38),
            method="continuous", moderator="pos1", moderator.value=c(1), add=FALSE, ylim=c(-.05,.22), ylab="Expected Topic
Proportion", xlab="Days since local camp established",
            xlim=c(-60,166), xaxt="n", linecol='red', printlegend=FALSE)
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(38),
```

```
            method="continuous", moderator="pos1", moderator.value=c(2), add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(38),
            method="continuous", moderator="pos1", moderator.value=c(3), add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 legend("topleft", c('Fewer Power Centers', 'More Power Centers', 'Most Power Centers'),
      lwd=2, col=c('red', 'blue', 'darkgreen'), bty="n", cex=1)


 #Camp Activity by size of # independent power centers
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(19),
            method="continuous", moderator="pos1", moderator.value=c(1), add=FALSE, ylim=c(-.2,.25), ylab="Expected Topic
Proportion", xlab="Days since local camp established",
            xlim=c(-60,166), xaxt="n", linecol='red', printlegend=FALSE)
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(19),
            method="continuous", moderator="pos1", moderator.value=c(2), add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(19),
            method="continuous", moderator="pos1", moderator.value=c(3), add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 mtext("Expected Topic Proportion", 2, line=4)


 #Weekday Marches by size of # independent power centers
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(5),
            method="continuous", moderator="pos1", moderator.value=c(1), add=FALSE, ylim=c(-.05,.2), ylab="Expected Topic
Proportion", xlab="Days since local camp established",
            xlim=c(-60,166), linecol='red', printlegend=FALSE)
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(5),
            method="continuous", moderator="pos1", moderator.value=c(2), add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(5),
            method="continuous", moderator="pos1", moderator.value=c(3), add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 mtext("Days Since Local Camp Established", 1, line=3)
}

pdf.f(j, file= '/Users/nickbadams/Dropbox/NickDiss/Chapter 4 Figures and Tables/pos_activity_w3.pdf',
    width=4, height=7)


k <- function(){
 layout(matrix(1:3, ncol=1, byrow=TRUE))
 par(oma=c(4,6,2,1), mar=c(1,1,1,1), mgp=c(2,1,0),
    cex.axis=1.5)
 #Rallies by size of # independent power centers
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(12),
            method="continuous", moderator="pos1", moderator.value=c(1), add=FALSE, ylim=c(-.1,.15), ylab="Expected Topic
Proportion", xaxt="n",
            xlim=c(-60,166), linecol='red', printlegend=FALSE)
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(12),
            method="continuous", moderator="pos1", moderator.value=c(2), add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(12),
            method="continuous", moderator="pos1", moderator.value=c(3), add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 legend("topleft", c('Fewer Power Centers', 'More Power Centers', 'Most Power Centers'),
      lwd=2, col=c('red', 'blue', 'darkgreen'), bty="n", cex=1)

 #Demonstrations by size of # independent power centers
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(1),
            method="continuous", moderator="pos1", moderator.value=c(1), add=FALSE, ylim=c(-.1,.125), ylab="Expected Topic
Proportion",  xaxt="n",
            xlim=c(-60,166), linecol='red', printlegend=FALSE)
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(1),
            method="continuous", moderator="pos1", moderator.value=c(2), add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(1),
```

```
            method="continuous", moderator="pos1", moderator.value=c(3), add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 mtext("Expected Topic Proportion", 2, line=4)


 #Labor Alliances by size of # independent power centers
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(35),
            method="continuous", moderator="pos1", moderator.value=c(1), add=FALSE, ylim=c(-.12,.125), ylab="Expected Topic
Proportion", xlab="Days since local camp established",
            xlim=c(-60,166), linecol='red', printlegend=FALSE)
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(35),
            method="continuous", moderator="pos1", moderator.value=c(2), add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(35),
            method="continuous", moderator="pos1", moderator.value=c(3), add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 mtext("Days Since Local Camp Established", 1, line=3)


}

pdf.f(k, file= '/Users/nickbadams/Dropbox/NickDiss/Chapter 4 Figures and Tables/pos_rallies_w3.pdf',
    width=4, height=7)


l <- function(){
 layout(matrix(1:3, ncol=1, byrow=TRUE))
 par(oma=c(4,6,2,1), mar=c(1,1,1,1), mgp=c(2,1,0),
   cex.axis=1.5)
 #Rallies by size of # independent power centers
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(27),
            method="continuous", moderator="pos1", moderator.value=c(1), add=FALSE, ylim=c(-.1,.15), ylab="Expected Topic
Proportion", xaxt="n",
            xlim=c(-60,166), linecol='red', printlegend=FALSE)
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(27),
            method="continuous", moderator="pos1", moderator.value=c(2), add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(27),
            method="continuous", moderator="pos1", moderator.value=c(3), add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 legend("topleft", c('Fewer Power Centers', 'More Power Centers', 'Most Power Centers'),
      lwd=2, col=c('red', 'blue', 'darkgreen'), bty="n", cex=1)

 #Demonstrations by size of # independent power centers
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(24),
            method="continuous", moderator="pos1", moderator.value=c(1), add=FALSE, ylim=c(-.1,.125), ylab="Expected Topic
Proportion",  xaxt="n",
            xlim=c(-60,166), linecol='red', printlegend=FALSE)
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(24),
            method="continuous", moderator="pos1", moderator.value=c(2), add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(24),
            method="continuous", moderator="pos1", moderator.value=c(3), add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 mtext("Expected Topic Proportion", 2, line=4)


 #Labor Alliances by size of # independent power centers
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(4),
            method="continuous", moderator="pos1", moderator.value=c(1), add=FALSE, ylim=c(-.12,.125), ylab="Expected Topic
Proportion", xlab="Days since local camp established",
            xlim=c(-60,166), linecol='red', printlegend=FALSE)
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(4),
            method="continuous", moderator="pos1", moderator.value=c(2), add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(4),
            method="continuous", moderator="pos1", moderator.value=c(3), add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 mtext("Days Since Local Camp Established", 1, line=3)
```

```
}

pdf.f(l, file= '/Users/nickbadams/Dropbox/NickDiss/Chapter 4 Figures and Tables/pos_targeting_w3.pdf',
    width=4, height=7)

m <- function(){
 layout(matrix(1:3, ncol=1, byrow=TRUE))
 par(oma=c(4,6,2,1), mar=c(1,1,1,1), mgp=c(2,1,0),
    cex.axis=1.5)
 #Rallies by size of # independent power centers
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(22),
                method="continuous", moderator="pos1", moderator.value=c(1), add=FALSE, ylim=c(-.1,.15), ylab="Expected Topic
Proportion", xaxt="n",
                xlim=c(-60,166), linecol='red', printlegend=FALSE)
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(22),
                method="continuous", moderator="pos1", moderator.value=c(2), add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(22),
                method="continuous", moderator="pos1", moderator.value=c(3), add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 legend("topleft", c('Fewer Power Centers', 'More Power Centers', 'Most Power Centers'),
      lwd=2, col=c('red', 'blue', 'darkgreen'), bty="n", cex=1)

 #Demonstrations by size of # independent power centers
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(29),
                method="continuous", moderator="pos1", moderator.value=c(1), add=FALSE, ylim=c(-.1,.125), ylab="Expected Topic
Proportion",  xaxt="n",
                xlim=c(-60,166), linecol='red', printlegend=FALSE)
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(29),
                method="continuous", moderator="pos1", moderator.value=c(2), add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(29),
                method="continuous", moderator="pos1", moderator.value=c(3), add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 mtext("Expected Topic Proportion", 2, line=4)


 #Labor Alliances by size of # independent power centers
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(7),
                method="continuous", moderator="pos1", moderator.value=c(1), add=FALSE, ylim=c(-.12,.125), ylab="Expected Topic
Proportion", xlab="Days since local camp established",
                xlim=c(-60,166), linecol='red', printlegend=FALSE)
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(7),
                method="continuous", moderator="pos1", moderator.value=c(2), add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(7),
                method="continuous", moderator="pos1", moderator.value=c(3), add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 mtext("Days Since Local Camp Established", 1, line=3)

}
pdf.f(m, file= '/Users/nickbadams/Dropbox/NickDiss/Chapter 4 Figures and Tables/pos_disrupt_w3.pdf',
    width=4, height=7)



#########################################################
#########################################################
#Activity by stability2
#########################################################
#########################################################

n <- function(){
 layout(matrix(1:3, ncol=1, byrow=TRUE))
 par(oma=c(4,6,2,1), mar=c(1,1,1,1), mgp=c(2,1,0),
    cex.axis=1.5)
 #Weekend Rallies by size of stability
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(38),
```

```
            method="continuous", moderator="stability2", moderator.value=c(1), add=FALSE, ylim=c(-.05,.22), ylab="Expected Topic
Proportion", xlab="Days since local camp established",
            xlim=c(-60,166), xaxt="n", linecol='red', printlegend=FALSE)
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(38),
            method="continuous", moderator="stability2", moderator.value=c(2), add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(38),
            method="continuous", moderator="stability2", moderator.value=c(3), add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 legend("topleft", c('More Political Stability', 'Less Political Stability', 'Least Political Stability'),
      lwd=2, col=c('red', 'blue', 'darkgreen'), bty="n", cex=1)


 #Camp Activity by size of stability
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(19),
            method="continuous", moderator="stability2", moderator.value=c(1), add=FALSE, ylim=c(-.2,.25), ylab="Expected Topic
Proportion", xlab="Days since local camp established",
            xlim=c(-60,166), xaxt="n", linecol='red', printlegend=FALSE)
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(19),
            method="continuous", moderator="stability2", moderator.value=c(2), add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(19),
            method="continuous", moderator="stability2", moderator.value=c(3), add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 mtext("Expected Topic Proportion", 2, line=4)


 #Weekday Marches by size of stability
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(5),
            method="continuous", moderator="stability2", moderator.value=c(1), add=FALSE, ylim=c(-.05,.2), ylab="Expected Topic
Proportion", xlab="Days since local camp established",
            xlim=c(-60,166), linecol='red', printlegend=FALSE)
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(5),
            method="continuous", moderator="stability2", moderator.value=c(2), add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(5),
            method="continuous", moderator="stability2", moderator.value=c(3), add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 mtext("Days Since Local Camp Established", 1, line=3)
}

pdf.f(n, file= '/Users/nickbadams/Dropbox/NickDiss/Chapter 4 Figures and Tables/stab_activity_w3.pdf',
    width=4, height=7)


o <- function(){
 layout(matrix(1:3, ncol=1, byrow=TRUE))
 par(oma=c(4,6,2,1), mar=c(1,1,1,1), mgp=c(2,1,0),
   cex.axis=1.5)
 #Rallies by size of stability
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(12),
            method="continuous", moderator="stability2", moderator.value=c(1), add=FALSE, ylim=c(-.1,.15), ylab="Expected Topic
Proportion", xaxt="n",
            xlim=c(-60,166), linecol='red', printlegend=FALSE)
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(12),
            method="continuous", moderator="stability2", moderator.value=c(2), add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(12),
            method="continuous", moderator="stability2", moderator.value=c(3), add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 legend("topleft", c('More Political Stability', 'Less Political Stability', 'Least Political Stability'),
      lwd=2, col=c('red', 'blue', 'darkgreen'), bty="n", cex=1)

 #Demonstrations by size of stability
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(1),
            method="continuous", moderator="stability2", moderator.value=c(1), add=FALSE, ylim=c(-.1,.125), ylab="Expected Topic
Proportion",  xaxt="n",
            xlim=c(-60,166), linecol='red', printlegend=FALSE)
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(1),
```

```
            method="continuous", moderator="stability2", moderator.value=c(2), add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(1),
            method="continuous", moderator="stability2", moderator.value=c(3), add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 mtext("Expected Topic Proportion", 2, line=4)


 #Labor Alliances by size of stability
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(35),
            method="continuous", moderator="stability2", moderator.value=c(1), add=FALSE, ylim=c(-.12,.125), ylab="Expected Topic
Proportion", xlab="Days since local camp established",
            xlim=c(-60,166), linecol='red', printlegend=FALSE)
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(35),
            method="continuous", moderator="stability2", moderator.value=c(2), add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(35),
            method="continuous", moderator="stability2", moderator.value=c(3), add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 mtext("Days Since Local Camp Established", 1, line=3)


}

pdf.f(o, file= '/Users/nickbadams/Dropbox/NickDiss/Chapter 4 Figures and Tables/stab_rallies_w3.pdf',
    width=4, height=7)


p <- function(){
 layout(matrix(1:3, ncol=1, byrow=TRUE))
 par(oma=c(4,6,2,1), mar=c(1,1,1,1), mgp=c(2,1,0),
    cex.axis=1.5)
 #Rallies by size of stability
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(27),
            method="continuous", moderator="stability2", moderator.value=c(1), add=FALSE, ylim=c(-.1,.15), ylab="Expected Topic
Proportion", xaxt="n",
            xlim=c(-60,166), linecol='red', printlegend=FALSE)
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(27),
            method="continuous", moderator="stability2", moderator.value=c(2), add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(27),
            method="continuous", moderator="stability2", moderator.value=c(3), add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 legend("topleft", c('More Political Stability', 'Less Political Stability', 'Least Political Stability'),
       lwd=2, col=c('red', 'blue', 'darkgreen'), bty="n", cex=1)

 #Demonstrations by size of stability
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(24),
            method="continuous", moderator="stability2", moderator.value=c(1), add=FALSE, ylim=c(-.1,.125), ylab="Expected Topic
Proportion",  xaxt="n",
            xlim=c(-60,166), linecol='red', printlegend=FALSE)
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(24),
            method="continuous", moderator="stability2", moderator.value=c(2), add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(24),
            method="continuous", moderator="stability2", moderator.value=c(3), add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 mtext("Expected Topic Proportion", 2, line=4)


 #Labor Alliances by size of stability
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(4),
            method="continuous", moderator="stability2", moderator.value=c(1), add=FALSE, ylim=c(-.12,.125), ylab="Expected Topic
Proportion", xlab="Days since local camp established",
            xlim=c(-60,166), linecol='red', printlegend=FALSE)
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(4),
            method="continuous", moderator="stability2", moderator.value=c(2), add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(4),
```

```
            method="continuous", moderator="stability2", moderator.value=c(3), add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 mtext("Days Since Local Camp Established", 1, line=3)

}

pdf.f(p, file= '/Users/nickbadams/Dropbox/NickDiss/Chapter 4 Figures and Tables/stab_targeting_w3.pdf',
    width=4, height=7)

q <- function(){
 layout(matrix(1:3, ncol=1, byrow=TRUE))
 par(oma=c(4,6,2,1), mar=c(1,1,1,1), mgp=c(2,1,0),
    cex.axis=1.5)
 #Rallies by size of stability
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(22),
            method="continuous", moderator="stability2", moderator.value=c(1), add=FALSE, ylim=c(-.1,.15), ylab="Expected Topic
Proportion", xaxt="n",
            xlim=c(-60,166), linecol='red', printlegend=FALSE)
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(22),
            method="continuous", moderator="stability2", moderator.value=c(2), add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(22),
            method="continuous", moderator="stability2", moderator.value=c(3), add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 legend("topleft", c('More Political Stability', 'Less Political Stability', 'Least Political Stability'),
      lwd=2, col=c('red', 'blue', 'darkgreen'), bty="n", cex=1)

 #Demonstrations by size of stability
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(29),
            method="continuous", moderator="stability2", moderator.value=c(1), add=FALSE, ylim=c(-.1,.125), ylab="Expected Topic
Proportion",  xaxt="n",
            xlim=c(-60,166), linecol='red', printlegend=FALSE)
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(29),
            method="continuous", moderator="stability2", moderator.value=c(2), add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(29),
            method="continuous", moderator="stability2", moderator.value=c(3), add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 mtext("Expected Topic Proportion", 2, line=4)


 #Labor Alliances by size of stability
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(7),
            method="continuous", moderator="stability2", moderator.value=c(1), add=FALSE, ylim=c(-.12,.125), ylab="Expected Topic
Proportion", xlab="Days since local camp established",
            xlim=c(-60,166), linecol='red', printlegend=FALSE)
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(7),
            method="continuous", moderator="stability2", moderator.value=c(2), add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(preplc40, covariate="dayscamped", model=lifecourse40,topic=c(7),
            method="continuous", moderator="stability2", moderator.value=c(3), add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 mtext("Days Since Local Camp Established", 1, line=3)

}
pdf.f(q, file= '/Users/nickbadams/Dropbox/NickDiss/Chapter 4 Figures and Tables/stab_disrupt_w3.pdf',
    width=4, height=7)
```

```
######plot writing function
pdf.f <- function(f, file, ...) {
  cat(sprintf('Writing %s\n', file))
  pdf(file, ...)
  on.exit(dev.off())
  f()
}



prep3int <- estimateEffect(c(38, 5, 19, 12, 1, 35, 27, 24, 4, 22, 29, 7) ~ pos1 + stability2 + pool2*s(dayscamped), lifecourse40xpool2,
                           metadata=meta, uncertainty="None")
#################
#################
#################


r <- function(){
  layout(matrix(1:3, ncol=1, byrow=TRUE))
  par(oma=c(4,6,2,1), mar=c(1,1,1,1), mgp=c(2,1,0),
     cex.axis=1.5)
plot.estimateEffect(prep3int, covariate="dayscamped", model=lifecourse40xpool2,topic=c(38),
                    method="continuous", moderator="pool2", xaxt='n', xlim=c(-60,166), moderator.value=1, add=FALSE, ylim=c(-.3,.5),
ylab="Expected Topic Proportion",
                   linecol='red', printlegend=FALSE)
plot.estimateEffect(prep3int, covariate="dayscamped", model=lifecourse40xpool2,topic=c(38),
                    method="continuous", moderator="pool2", xaxt='n',moderator.value=c(3), add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
plot.estimateEffect(prep3int, covariate="dayscamped", model=lifecourse40xpool2,topic=c(38),
                    method="continuous", moderator="pool2", moderator.value=c(5), add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
legend("top", c('Smaller, Less Liberal Population', 'Modal Population', 'Large, Liberal Population'),
       lwd=2, col=c('red', 'blue', 'darkgreen'), bty="n")

#Topic 19 "Camp Activities"
plot.estimateEffect(prep3int, covariate="dayscamped", model=lifecourse40xpool2,topic=c(19),
                    method="continuous", moderator="pool2", moderator.value=1, add=FALSE, ylim=c(-.3,.5), ylab="Expected Topic Proportion",
xlab="Days since local camp established",
                    xlim=c(-60,166), xaxt='n', linecol='red', printlegend=FALSE)
plot.estimateEffect(prep3int, covariate="dayscamped", model=lifecourse40xpool2,topic=c(19),
                    method="continuous", moderator="pool2", moderator.value=c(3), add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
plot.estimateEffect(prep3int, covariate="dayscamped", model=lifecourse40xpool2,topic=c(19),
                    method="continuous", moderator="pool2", moderator.value=c(5), add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
              mtext("Expected Topic Proportion", 2, line=4)

plot.estimateEffect(prep3int, covariate="dayscamped", model=lifecourse40xpool2,topic=c(5),
                    method="continuous", moderator="pool2", moderator.value=1, add=FALSE, ylim=c(-.3,.4), ylab="Expected Topic Proportion",
xlab="Days since local camp established",
                    xlim=c(-60,166), linecol='red', printlegend=FALSE)
plot.estimateEffect(prep3int, covariate="dayscamped", model=lifecourse40xpool2,topic=c(5),
                    method="continuous", moderator="pool2", moderator.value=c(3), add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
plot.estimateEffect(prep3int, covariate="dayscamped", model=lifecourse40xpool2,topic=c(5),
                    method="continuous", moderator="pool2", moderator.value=c(5), add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
           mtext("Days Since Local Camp Established", 1, line=3)

}
pdf.f(r, file= '/Users/nickbadams/Dropbox/NickDiss/Chapter 4 Figures and Tables/intpool_activities_w3.pdf',
      width=4, height=7)
####################
####################
####################
####################

rr <- function(){
  layout(matrix(1:3, ncol=1, byrow=TRUE))
```

```
 par(oma=c(4,6,2,1), mar=c(1,1,1,1), mgp=c(2,1,0),
    cex.axis=1.5)
 plot.estimateEffect(prep3int, covariate="dayscamped", model=lifecourse40xpool2,topic=c(12),
                method="continuous", moderator="pool2", xaxt='n', xlim=c(-60,166), moderator.value=1, add=FALSE, ylim=c(-.3,.5),
ylab="Expected Topic Proportion",
                linecol='red', printlegend=FALSE)
 plot.estimateEffect(prep3int, covariate="dayscamped", model=lifecourse40xpool2,topic=c(12),
                method="continuous", moderator="pool2", xaxt='n',moderator.value=c(3), add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(prep3int, covariate="dayscamped", model=lifecourse40xpool2,topic=c(12),
                method="continuous", moderator="pool2", moderator.value=c(5), add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 legend("top", c('Smaller, Less Liberal Population', 'Modal Population', 'Large, Liberal Population'),
       lwd=2, col=c('red', 'blue', 'darkgreen'), bty="n")

 #Topic 19 "Camp Activities"
 plot.estimateEffect(prep3int, covariate="dayscamped", model=lifecourse40xpool2,topic=c(1),
                method="continuous", moderator="pool2", moderator.value=1, add=FALSE, ylim=c(-.3,.5), ylab="Expected Topic Proportion",
xlab="Days since local camp established",
                xlim=c(-60,166), xaxt='n', linecol='red', printlegend=FALSE)
 plot.estimateEffect(prep3int, covariate="dayscamped", model=lifecourse40xpool2,topic=c(1),
                method="continuous", moderator="pool2", moderator.value=c(3), add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(prep3int, covariate="dayscamped", model=lifecourse40xpool2,topic=c(1),
                method="continuous", moderator="pool2", moderator.value=c(5), add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 mtext("Expected Topic Proportion", 2, line=4)

 plot.estimateEffect(prep3int, covariate="dayscamped", model=lifecourse40xpool2,topic=c(35),
                method="continuous", moderator="pool2", moderator.value=1, add=FALSE, ylim=c(-.3,.4), ylab="Expected Topic Proportion",
xlab="Days since local camp established",
                xlim=c(-60,166), linecol='red', printlegend=FALSE)
 plot.estimateEffect(prep3int, covariate="dayscamped", model=lifecourse40xpool2,topic=c(35),
                method="continuous", moderator="pool2", moderator.value=c(3), add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(prep3int, covariate="dayscamped", model=lifecourse40xpool2,topic=c(35),
                method="continuous", moderator="pool2", moderator.value=c(5), add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 mtext("Days Since Local Camp Established", 1, line=3)

}
pdf.f(rr, file= '/Users/nickbadams/Dropbox/NickDiss/Chapter 4 Figures and Tables/intpool_rallies_w3.pdf',
    width=4, height=7)

###################
###################
###################
###################
t <- function(){
 layout(matrix(1:3, ncol=1, byrow=TRUE))
 par(oma=c(4,6,2,1), mar=c(1,1,1,1), mgp=c(2,1,0),
    cex.axis=1.5)
 plot.estimateEffect(prep3int, covariate="dayscamped", model=lifecourse40xpool2,topic=c(27),
                method="continuous", moderator="pool2", xaxt='n', xlim=c(-60,166), moderator.value=1, add=FALSE, ylim=c(-.3,.5),
ylab="Expected Topic Proportion",
                linecol='red', printlegend=FALSE)
 plot.estimateEffect(prep3int, covariate="dayscamped", model=lifecourse40xpool2,topic=c(27),
                method="continuous", moderator="pool2", xaxt='n',moderator.value=c(3), add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(prep3int, covariate="dayscamped", model=lifecourse40xpool2,topic=c(27),
                method="continuous", moderator="pool2", moderator.value=c(5), add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 legend("top", c('Smaller, Less Liberal Population', 'Modal Population', 'Large, Liberal Population'),
       lwd=2, col=c('red', 'blue', 'darkgreen'), bty="n")

 plot.estimateEffect(prep3int, covariate="dayscamped", model=lifecourse40xpool2,topic=c(24),
                method="continuous", moderator="pool2", moderator.value=1, add=FALSE, ylim=c(-.3,.5), ylab="Expected Topic Proportion",
xlab="Days since local camp established",
                xlim=c(-60,166), xaxt='n', linecol='red', printlegend=FALSE)
```

```
 plot.estimateEffect(prep3int, covariate="dayscamped", model=lifecourse40xpool2,topic=c(24),
                method="continuous", moderator="pool2", moderator.value=c(3), add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(prep3int, covariate="dayscamped", model=lifecourse40xpool2,topic=c(24),
                method="continuous", moderator="pool2", moderator.value=c(5), add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 mtext("Expected Topic Proportion", 2, line=4)

 plot.estimateEffect(prep3int, covariate="dayscamped", model=lifecourse40xpool2,topic=c(4),
                method="continuous", moderator="pool2", moderator.value=1, add=FALSE, ylim=c(-.3,.4), ylab="Expected Topic Proportion",
xlab="Days since local camp established",
                xlim=c(-60,166), linecol='red', printlegend=FALSE)
 plot.estimateEffect(prep3int, covariate="dayscamped", model=lifecourse40xpool2,topic=c(4),
                method="continuous", moderator="pool2", moderator.value=c(3), add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(prep3int, covariate="dayscamped", model=lifecourse40xpool2,topic=c(4),
                method="continuous", moderator="pool2", moderator.value=c(5), add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 mtext("Days Since Local Camp Established", 1, line=3)

}
pdf.f(t, file= '/Users/nickbadams/Dropbox/NickDiss/Chapter 4 Figures and Tables/intpool_targeting_w3.pdf',
    width=4, height=7)

####################
####################
####################
####################
####################
####################
####################
u <- function(){
 layout(matrix(1:3, ncol=1, byrow=TRUE))
 par(oma=c(4,6,2,1), mar=c(1,1,1,1), mgp=c(2,1,0),
   cex.axis=1.5)
 plot.estimateEffect(prep3int, covariate="dayscamped", model=lifecourse40xpool2,topic=c(22),
                method="continuous", moderator="pool2", xaxt='n', xlim=c(-60,166), moderator.value=1, add=FALSE, ylim=c(-.3,.5),
ylab="Expected Topic Proportion",
                linecol='red', printlegend=FALSE)
 plot.estimateEffect(prep3int, covariate="dayscamped", model=lifecourse40xpool2,topic=c(22),
                method="continuous", moderator="pool2", xaxt='n',moderator.value=c(3), add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(prep3int, covariate="dayscamped", model=lifecourse40xpool2,topic=c(22),
                method="continuous", moderator="pool2", moderator.value=c(5), add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 legend("top", c('Smaller, Less Liberal Population', 'Modal Population', 'Large, Liberal Population'),
      lwd=2, col=c('red', 'blue', 'darkgreen'), bty="n")

 plot.estimateEffect(prep3int, covariate="dayscamped", model=lifecourse40xpool2,topic=c(29),
                method="continuous", moderator="pool2", moderator.value=1, add=FALSE, ylim=c(-.3,.5), ylab="Expected Topic Proportion",
xlab="Days since local camp established",
                xlim=c(-60,166), xaxt='n', linecol='red', printlegend=FALSE)
 plot.estimateEffect(prep3int, covariate="dayscamped", model=lifecourse40xpool2,topic=c(29),
                method="continuous", moderator="pool2", moderator.value=c(3), add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(prep3int, covariate="dayscamped", model=lifecourse40xpool2,topic=c(29),
                method="continuous", moderator="pool2", moderator.value=c(5), add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 mtext("Expected Topic Proportion", 2, line=4)

 plot.estimateEffect(prep3int, covariate="dayscamped", model=lifecourse40xpool2,topic=c(7),
                method="continuous", moderator="pool2", moderator.value=1, add=FALSE, ylim=c(-.3,.4), ylab="Expected Topic Proportion",
xlab="Days since local camp established",
                xlim=c(-60,166), linecol='red', printlegend=FALSE)
 plot.estimateEffect(prep3int, covariate="dayscamped", model=lifecourse40xpool2,topic=c(7),
                method="continuous", moderator="pool2", moderator.value=c(3), add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(prep3int, covariate="dayscamped", model=lifecourse40xpool2,topic=c(7),
```

```
            method="continuous", moderator="pool2", moderator.value=c(5), add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 mtext("Days Since Local Camp Established", 1, line=3)

}
pdf.f(u, file= '/Users/nickbadams/Dropbox/NickDiss/Chapter 4 Figures and Tables/intpool_disrupt_w3.pdf',
    width=4, height=7)




#####################################
#####################################

lifecourse40xstability2 <- stm(out$documents,out$vocab,K=40,prevalence =~ pos1 + stability2*s(dayscamped) + pool2, max.em.its=250,
data=out$meta, seed=4813083)
save(lifecourse40xstability2,file="STMlf40xstability2.Rda")

prep4int <- estimateEffect(c(38, 5, 19, 12, 1, 35, 27, 24, 4, 22, 29, 7) ~ pos1 + stability2*s(dayscamped) + pool2, lifecourse40xstability2,
                  metadata=meta, uncertainty="None")
#################
#################

#################

#################
#              ###########       STABILITY INTERACTIONS    ##############################################
#################


v <- function(){
 layout(matrix(1:3, ncol=1, byrow=TRUE))
 par(oma=c(4,6,2,1), mar=c(1,1,1,1), mgp=c(2,1,0),
    cex.axis=1.5)
 plot.estimateEffect(prep4int, covariate="dayscamped", model=lifecourse40xstability2,topic=c(38),
            method="continuous", moderator="stability2", xaxt='n', xlim=c(-60,166), moderator.value=1, add=FALSE, ylim=c(-.3,.5),
ylab="Expected Topic Proportion",
            linecol='red', printlegend=FALSE)
 plot.estimateEffect(prep4int, covariate="dayscamped", model=lifecourse40xstability2,topic=c(38),
            method="continuous", moderator="stability2", xaxt='n',moderator.value=2, add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(prep4int, covariate="dayscamped", model=lifecourse40xstability2,topic=c(38),
            method="continuous", moderator="stability2", moderator.value=3, add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 legend("top", c('Stable', 'Less Stable', 'Least Stable'),
      lwd=2, col=c('red', 'blue', 'darkgreen'), bty="n")

 #Topic 19 "Camp Activities"
 plot.estimateEffect(prep4int, covariate="dayscamped", model=lifecourse40xstability2,topic=c(19),
            method="continuous", moderator="stability2", moderator.value=1, add=FALSE, ylim=c(-.3,.5), ylab="Expected Topic
Proportion", xlab="Days since local camp established",
            xlim=c(-60,166), xaxt='n', linecol='red', printlegend=FALSE)
 plot.estimateEffect(prep4int, covariate="dayscamped", model=lifecourse40xstability2,topic=c(19),
            method="continuous", moderator="stability2", moderator.value=2, add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(prep4int, covariate="dayscamped", model=lifecourse40xstability2,topic=c(19),
            method="continuous", moderator="stability2", moderator.value=3, add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 mtext("Expected Topic Proportion", 2, line=4)

 plot.estimateEffect(prep4int, covariate="dayscamped", model=lifecourse40xstability2,topic=c(5),
            method="continuous", moderator="stability2", moderator.value=1, add=FALSE, ylim=c(-.3,.4), ylab="Expected Topic
Proportion", xlab="Days since local camp established",
            xlim=c(-60,166), linecol='red', printlegend=FALSE)
 plot.estimateEffect(prep4int, covariate="dayscamped", model=lifecourse40xstability2,topic=c(5),
            method="continuous", moderator="stability2", moderator.value=2, add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(prep4int, covariate="dayscamped", model=lifecourse40xstability2,topic=c(5),
```

```
            method="continuous", moderator="stability2", moderator.value=3, add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 mtext("Days Since Local Camp Established", 1, line=3)

}
pdf.f(v, file= '/Users/nickbadams/Dropbox/NickDiss/Chapter 4 Figures and Tables/intstability_activities.pdf',
    width=4, height=7)
####################
####################
####################
####################

w <- function(){
 layout(matrix(1:3, ncol=1, byrow=TRUE))
 par(oma=c(4,6,2,1), mar=c(1,1,1,1), mgp=c(2,1,0),
    cex.axis=1.5)
 plot.estimateEffect(prep4int, covariate="dayscamped", model=lifecourse40xstability2,topic=c(12),
            method="continuous", moderator="stability2", xaxt='n', xlim=c(-60,166), moderator.value=1, add=FALSE, ylim=c(-.3,.5),
ylab="Expected Topic Proportion",
            linecol='red', printlegend=FALSE)
 plot.estimateEffect(prep4int, covariate="dayscamped", model=lifecourse40xstability2,topic=c(12),
            method="continuous", moderator="stability2", xaxt='n',moderator.value=2, add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(prep4int, covariate="dayscamped", model=lifecourse40xstability2,topic=c(12),
            method="continuous", moderator="stability2", moderator.value=3, add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 legend("top", c('Stable', 'Less Stable', 'Least Stable'),
      lwd=2, col=c('red', 'blue', 'darkgreen'), bty="n")

 #Topic 19 "Camp Activities"
 plot.estimateEffect(prep4int, covariate="dayscamped", model=lifecourse40xstability2,topic=c(1),
            method="continuous", moderator="stability2", moderator.value=1, add=FALSE, ylim=c(-.3,.5), ylab="Expected Topic
Proportion", xlab="Days since local camp established",
            xlim=c(-60,166), xaxt='n', linecol='red', printlegend=FALSE)
 plot.estimateEffect(prep4int, covariate="dayscamped", model=lifecourse40xstability2,topic=c(1),
            method="continuous", moderator="stability2", moderator.value=2, add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(prep4int, covariate="dayscamped", model=lifecourse40xstability2,topic=c(1),
            method="continuous", moderator="stability2", moderator.value=3, add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 mtext("Expected Topic Proportion", 2, line=4)

 plot.estimateEffect(prep4int, covariate="dayscamped", model=lifecourse40xstability2,topic=c(35),
            method="continuous", moderator="stability2", moderator.value=1, add=FALSE, ylim=c(-.3,.4), ylab="Expected Topic
Proportion", xlab="Days since local camp established",
            xlim=c(-60,166), linecol='red', printlegend=FALSE)
 plot.estimateEffect(prep4int, covariate="dayscamped", model=lifecourse40xstability2,topic=c(35),
            method="continuous", moderator="stability2", moderator.value=2, add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(prep4int, covariate="dayscamped", model=lifecourse40xstability2,topic=c(35),
            method="continuous", moderator="stability2", moderator.value=3, add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 mtext("Days Since Local Camp Established", 1, line=3)

}
pdf.f(w, file= '/Users/nickbadams/Dropbox/NickDiss/Chapter 4 Figures and Tables/intStability_rallies.pdf',
    width=4, height=7)

####################
####################
####################
####################
x <- function(){
 layout(matrix(1:3, ncol=1, byrow=TRUE))
 par(oma=c(4,6,2,1), mar=c(1,1,1,1), mgp=c(2,1,0),
    cex.axis=1.5)
 plot.estimateEffect(prep4int, covariate="dayscamped", model=lifecourse40xstability2,topic=c(27),
            method="continuous", moderator="stability2", xaxt='n', xlim=c(-60,166), moderator.value=1, add=FALSE, ylim=c(-.3,.5),
ylab="Expected Topic Proportion",
```

```
            linecol='red', printlegend=FALSE)
 plot.estimateEffect(prep4int, covariate="dayscamped", model=lifecourse40xstability2,topic=c(27),
            method="continuous", moderator="stability2", xaxt='n',moderator.value=2, add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(prep4int, covariate="dayscamped", model=lifecourse40xstability2,topic=c(27),
            method="continuous", moderator="stability2", moderator.value=3, add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 legend("top", c('Stable', 'Less Stable', 'Least Stable'),
      lwd=2, col=c('red', 'blue', 'darkgreen'), bty="n")

 plot.estimateEffect(prep4int, covariate="dayscamped", model=lifecourse40xstability2,topic=c(24),
            method="continuous", moderator="stability2", moderator.value=1, add=FALSE, ylim=c(-.3,.5), ylab="Expected Topic
Proportion", xlab="Days since local camp established",
            xlim=c(-60,166), xaxt='n', linecol='red', printlegend=FALSE)
 plot.estimateEffect(prep4int, covariate="dayscamped", model=lifecourse40xstability2,topic=c(24),
            method="continuous", moderator="stability2", moderator.value=2, add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(prep4int, covariate="dayscamped", model=lifecourse40xstability2,topic=c(24),
            method="continuous", moderator="stability2", moderator.value=3, add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 mtext("Expected Topic Proportion", 2, line=4)

 plot.estimateEffect(prep4int, covariate="dayscamped", model=lifecourse40xstability2,topic=c(4),
            method="continuous", moderator="stability2", moderator.value=1, add=FALSE, ylim=c(-.3,.4), ylab="Expected Topic
Proportion", xlab="Days since local camp established",
            xlim=c(-60,166), linecol='red', printlegend=FALSE)
 plot.estimateEffect(prep4int, covariate="dayscamped", model=lifecourse40xstability2,topic=c(4),
            method="continuous", moderator="stability2", moderator.value=2, add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(prep4int, covariate="dayscamped", model=lifecourse40xstability2,topic=c(4),
            method="continuous", moderator="stability2", moderator.value=3, add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 mtext("Days Since Local Camp Established", 1, line=3)

}
pdf.f(x, file= '/Users/nickbadams/Dropbox/NickDiss/Chapter 4 Figures and Tables/intStability_targeting.pdf',
    width=4, height=7)

####################
####################
####################
####################
####################
####################
####################
y <- function(){
 layout(matrix(1:3, ncol=1, byrow=TRUE))
 par(oma=c(4,6,2,1), mar=c(1,1,1,1), mgp=c(2,1,0),
    cex.axis=1.5)
 plot.estimateEffect(prep4int, covariate="dayscamped", model=lifecourse40xstability2,topic=c(22),
            method="continuous", moderator="stability2", xaxt='n', xlim=c(-60,166), moderator.value=1, add=FALSE, ylim=c(-.3,.5),
ylab="Expected Topic Proportion",
            linecol='red', printlegend=FALSE)
 plot.estimateEffect(prep4int, covariate="dayscamped", model=lifecourse40xstability2,topic=c(22),
            method="continuous", moderator="stability2", xaxt='n',moderator.value=2, add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(prep4int, covariate="dayscamped", model=lifecourse40xstability2,topic=c(22),
            method="continuous", moderator="stability2", moderator.value=3, add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 legend("top", c('Stable', 'Less Stable', 'Least Stable'),
      lwd=2, col=c('red', 'blue', 'darkgreen'), bty="n")

 plot.estimateEffect(prep4int, covariate="dayscamped", model=lifecourse40xstability2,topic=c(29),
            method="continuous", moderator="stability2", moderator.value=1, add=FALSE, ylim=c(-.3,.5), ylab="Expected Topic
Proportion", xlab="Days since local camp established",
            xlim=c(-60,166), xaxt='n', linecol='red', printlegend=FALSE)
 plot.estimateEffect(prep4int, covariate="dayscamped", model=lifecourse40xstability2,topic=c(29),
            method="continuous", moderator="stability2", moderator.value=2, add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
```

```
 plot.estimateEffect(prep4int, covariate="dayscamped", model=lifecourse40xstability2,topic=c(29),
              method="continuous", moderator="stability2", moderator.value=3, add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 mtext("Expected Topic Proportion", 2, line=4)

 plot.estimateEffect(prep4int, covariate="dayscamped", model=lifecourse40xstability2,topic=c(7),
              method="continuous", moderator="stability2", moderator.value=1, add=FALSE, ylim=c(-.3,.4), ylab="Expected Topic
Proportion", xlab="Days since local camp established",
              xlim=c(-60,166), linecol='red', printlegend=FALSE)
 plot.estimateEffect(prep4int, covariate="dayscamped", model=lifecourse40xstability2,topic=c(7),
              method="continuous", moderator="stability2", moderator.value=2, add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(prep4int, covariate="dayscamped", model=lifecourse40xstability2,topic=c(7),
              method="continuous", moderator="stability2", moderator.value=3, add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 mtext("Days Since Local Camp Established", 1, line=3)

}
pdf.f(y, file= '/Users/nickbadams/Dropbox/NickDiss/Chapter 4 Figures and Tables/intStability_disrupt.pdf',
    width=4, height=7)




####################################
####################################
####################################
####################################



####################################
####################################
lifecourse40xpos12 <- stm(out$documents,out$vocab,K=40,prevalence =~ pos1*s(dayscamped) + stability2 + pool2, max.em.its=250,
data=out$meta, seed=4813083)
save(lifecourse40xpos12,file="STMlf40xpos12.Rda")
####################################
####################################
load("STMlf40xpos12.Rda")


prep5int <- estimateEffect(c(38, 5, 19, 12, 1, 35, 27, 24, 4, 22, 29, 7) ~ pos1*s(dayscamped) + stability2 + pool2, lifecourse40xpos12,
                metadata=meta, uncertainty="None")
#################
#################

#################

#################
#              ###########       POS 1 Interactions   ##############################################
#################


z <- function(){
 layout(matrix(1:3, ncol=1, byrow=TRUE))
 par(oma=c(4,6,2,1), mar=c(1,1,1,1), mgp=c(2,1,0),
   cex.axis=1.5)
 plot.estimateEffect(prep5int, covariate="dayscamped", model=lifecourse40xpos12,topic=c(38),
              method="continuous", moderator="pos1", xaxt='n', xlim=c(-60,166), moderator.value=1, add=FALSE, ylim=c(-.3,.5),
ylab="Expected Topic Proportion",
              linecol='red', printlegend=FALSE)
 plot.estimateEffect(prep5int, covariate="dayscamped", model=lifecourse40xpos12,topic=c(38),
              method="continuous", moderator="pos1", xaxt='n',moderator.value=2, add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(prep5int, covariate="dayscamped", model=lifecourse40xpos12,topic=c(38),
              method="continuous", moderator="pos1", moderator.value=3, add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 legend("top", c('Fewest Centers of Power', 'More Centers of Power', 'Most Centers of Power'),
```

```
      lwd=2, col=c('red', 'blue', 'darkgreen'), bty="n")


 plot.estimateEffect(prep5int, covariate="dayscamped", model=lifecourse40xpos12,topic=c(19),
                method="continuous", moderator="pos1", moderator.value=1, add=FALSE, ylim=c(-.3,.5), ylab="Expected Topic Proportion",
xlab="Days since local camp established",
                xlim=c(-60,166), xaxt='n', linecol='red', printlegend=FALSE)
 plot.estimateEffect(prep5int, covariate="dayscamped", model=lifecourse40xpos12,topic=c(19),
                method="continuous", moderator="pos1", moderator.value=2, add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(prep5int, covariate="dayscamped", model=lifecourse40xpos12,topic=c(19),
                method="continuous", moderator="pos1", moderator.value=3, add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 mtext("Expected Topic Proportion", 2, line=4)

 plot.estimateEffect(prep5int, covariate="dayscamped", model=lifecourse40xpos12,topic=c(5),
                method="continuous", moderator="pos1", moderator.value=1, add=FALSE, ylim=c(-.3,.4), ylab="Expected Topic Proportion",
xlab="Days since local camp established",
                xlim=c(-60,166), linecol='red', printlegend=FALSE)
 plot.estimateEffect(prep5int, covariate="dayscamped", model=lifecourse40xpos12,topic=c(5),
                method="continuous", moderator="pos1", moderator.value=2, add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(prep5int, covariate="dayscamped", model=lifecourse40xpos12,topic=c(5),
                method="continuous", moderator="pos1", moderator.value=3, add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 mtext("Days Since Local Camp Established", 1, line=3)

}
pdf.f(z, file= '/Users/nickbadams/Dropbox/NickDiss/Chapter 4 Figures and Tables/intPos1_activities_fixed.pdf',
    width=4, height=7)
####################
####################
####################
####################

aa <- function(){
 layout(matrix(1:3, ncol=1, byrow=TRUE))
 par(oma=c(4,6,2,1), mar=c(1,1,1,1), mgp=c(2,1,0),
    cex.axis=1.5)
 plot.estimateEffect(prep5int, covariate="dayscamped", model=lifecourse40xpos12,topic=c(12),
                method="continuous", moderator="pos1", xaxt='n', xlim=c(-60,166), moderator.value=1, add=FALSE, ylim=c(-.3,.5),
ylab="Expected Topic Proportion",
                linecol='red', printlegend=FALSE)
 plot.estimateEffect(prep5int, covariate="dayscamped", model=lifecourse40xpos12,topic=c(12),
                method="continuous", moderator="pos1", xaxt='n',moderator.value=2, add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(prep5int, covariate="dayscamped", model=lifecourse40xpos12,topic=c(12),
                method="continuous", moderator="pos1", moderator.value=3, add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 legend("top", c('Fewest Centers of Power', 'More Centers of Power', 'Most Centers of Power'),
      lwd=2, col=c('red', 'blue', 'darkgreen'), bty="n")

 #Topic 19 "Camp Activities"
 plot.estimateEffect(prep5int, covariate="dayscamped", model=lifecourse40xpos12,topic=c(1),
                method="continuous", moderator="pos1", moderator.value=1, add=FALSE, ylim=c(-.3,.5), ylab="Expected Topic Proportion",
xlab="Days since local camp established",
                xlim=c(-60,166), xaxt='n', linecol='red', printlegend=FALSE)
 plot.estimateEffect(prep5int, covariate="dayscamped", model=lifecourse40xpos12,topic=c(1),
                method="continuous", moderator="pos1", moderator.value=2, add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(prep5int, covariate="dayscamped", model=lifecourse40xpos12,topic=c(1),
                method="continuous", moderator="pos1", moderator.value=3, add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 mtext("Expected Topic Proportion", 2, line=4)

 plot.estimateEffect(prep5int, covariate="dayscamped", model=lifecourse40xpos12,topic=c(35),
                method="continuous", moderator="pos1", moderator.value=1, add=FALSE, ylim=c(-.3,.4), ylab="Expected Topic Proportion",
xlab="Days since local camp established",
                xlim=c(-60,166), linecol='red', printlegend=FALSE)
```

```
 plot.estimateEffect(prep5int, covariate="dayscamped", model=lifecourse40xpos12,topic=c(35),
                method="continuous", moderator="pos1", moderator.value=2, add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(prep5int, covariate="dayscamped", model=lifecourse40xpos12,topic=c(35),
                method="continuous", moderator="pos1", moderator.value=3, add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 mtext("Days Since Local Camp Established", 1, line=3)

}
pdf.f(aa, file= '/Users/nickbadams/Dropbox/NickDiss/Chapter 4 Figures and Tables/intPos1_rallies_fixed.pdf',
    width=4, height=7)

###################
###################
###################
###################
bb <- function(){
 layout(matrix(1:3, ncol=1, byrow=TRUE))
 par(oma=c(4,6,2,1), mar=c(1,1,1,1), mgp=c(2,1,0),
   cex.axis=1.5)
 plot.estimateEffect(prep5int, covariate="dayscamped", model=lifecourse40xpos12,topic=c(27),
                method="continuous", moderator="pos1", xaxt='n', xlim=c(-60,166), moderator.value=1, add=FALSE, ylim=c(-.3,.5),
ylab="Expected Topic Proportion",
                linecol='red', printlegend=FALSE)
 plot.estimateEffect(prep5int, covariate="dayscamped", model=lifecourse40xpos12,topic=c(27),
                method="continuous", moderator="pos1", xaxt='n',moderator.value=2, add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(prep5int, covariate="dayscamped", model=lifecourse40xpos12,topic=c(27),
                method="continuous", moderator="pos1", moderator.value=3, add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 legend("top", c('Fewest Centers of Power', 'More Centers of Power', 'Most Centers of Power'),
      lwd=2, col=c('red', 'blue', 'darkgreen'), bty="n")

 plot.estimateEffect(prep5int, covariate="dayscamped", model=lifecourse40xpos12,topic=c(24),
                method="continuous", moderator="pos1", moderator.value=1, add=FALSE, ylim=c(-.3,.5), ylab="Expected Topic Proportion",
xlab="Days since local camp established",
                xlim=c(-60,166), xaxt='n', linecol='red', printlegend=FALSE)
 plot.estimateEffect(prep5int, covariate="dayscamped", model=lifecourse40xpos12,topic=c(24),
                method="continuous", moderator="pos1", moderator.value=2, add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(prep5int, covariate="dayscamped", model=lifecourse40xpos12,topic=c(24),
                method="continuous", moderator="pos1", moderator.value=3, add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 mtext("Expected Topic Proportion", 2, line=4)

 plot.estimateEffect(prep5int, covariate="dayscamped", model=lifecourse40xpos12,topic=c(4),
                method="continuous", moderator="pos1", moderator.value=1, add=FALSE, ylim=c(-.3,.4), ylab="Expected Topic Proportion",
xlab="Days since local camp established",
                xlim=c(-60,166), linecol='red', printlegend=FALSE)
 plot.estimateEffect(prep5int, covariate="dayscamped", model=lifecourse40xpos12,topic=c(4),
                method="continuous", moderator="pos1", moderator.value=2, add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(prep5int, covariate="dayscamped", model=lifecourse40xpos12,topic=c(4),
                method="continuous", moderator="pos1", moderator.value=3, add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 mtext("Days Since Local Camp Established", 1, line=3)

}
pdf.f(bb, file= '/Users/nickbadams/Dropbox/NickDiss/Chapter 4 Figures and Tables/intPos1_targeting_fixed.pdf',
    width=4, height=7)

###################
###################
###################
###################
###################
###################
###################
cc <- function(){
```

```
 layout(matrix(1:3, ncol=1, byrow=TRUE))
 par(oma=c(4,6,2,1), mar=c(1,1,1,1), mgp=c(2,1,0),
    cex.axis=1.5)
 plot.estimateEffect(prep5int, covariate="dayscamped", model=lifecourse40xpos12,topic=c(22),
              method="continuous", moderator="pos1", xaxt='n', xlim=c(-60,166), moderator.value=1, add=FALSE, ylim=c(-.3,.5),
ylab="Expected Topic Proportion",
              linecol='red', printlegend=FALSE)
 plot.estimateEffect(prep5int, covariate="dayscamped", model=lifecourse40xpos12,topic=c(22),
              method="continuous", moderator="pos1", xaxt='n',moderator.value=2, add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(prep5int, covariate="dayscamped", model=lifecourse40xpos12,topic=c(22),
              method="continuous", moderator="pos1", moderator.value=3, add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 legend("top", c('Fewest Centers of Power', 'More Centers of Power', 'Most Centers of Power'),
       lwd=2, col=c('red', 'blue', 'darkgreen'), bty="n")

 plot.estimateEffect(prep5int, covariate="dayscamped", model=lifecourse40xpos12,topic=c(29),
              method="continuous", moderator="pos1", moderator.value=1, add=FALSE, ylim=c(-.3,.5), ylab="Expected Topic Proportion",
xlab="Days since local camp established",
              xlim=c(-60,166), xaxt='n', linecol='red', printlegend=FALSE)
 plot.estimateEffect(prep5int, covariate="dayscamped", model=lifecourse40xpos12,topic=c(29),
              method="continuous", moderator="pos1", moderator.value=2, add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(prep5int, covariate="dayscamped", model=lifecourse40xpos12,topic=c(29),
              method="continuous", moderator="pos1", moderator.value=3, add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 mtext("Expected Topic Proportion", 2, line=4)

 plot.estimateEffect(prep5int, covariate="dayscamped", model=lifecourse40xpos12,topic=c(7),
              method="continuous", moderator="pos1", moderator.value=1, add=FALSE, ylim=c(-.3,.4), ylab="Expected Topic Proportion",
xlab="Days since local camp established",
              xlim=c(-60,166), linecol='red', printlegend=FALSE)
 plot.estimateEffect(prep5int, covariate="dayscamped", model=lifecourse40xpos12,topic=c(7),
              method="continuous", moderator="pos1", moderator.value=2, add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(prep5int, covariate="dayscamped", model=lifecourse40xpos12,topic=c(7),
              method="continuous", moderator="pos1", moderator.value=3, add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 mtext("Days Since Local Camp Established", 1, line=3)

}
pdf.f(cc, file= '/Users/nickbadams/Dropbox/NickDiss/Chapter 4 Figures and Tables/intPos1_disrupt_fixed.pdf',
    width=4, height=7)



####################################
####################################
####################################
```

Data Preparation and Topic Modeling in R – Chapter 6

The following is the R code used to perform structural topic modeling on police event text units. It starts by reading in data where each row is an event text unit. Then, data describing the cities in which events occurred are merged with the dataset. Some variables are generated. A date-of-event variable is derived from the article_date and the days (e.g. 'Tuesday') mentioned in the text. Continuous police department variables describing Police Capacity and Culture converted into ordinal variables. Then, Structural Topic Models are estimated and plotted.

```
rm(list=ls())
options(encoding= "UTF-8")
library(jsonlite)
library(plyr)
# #install.packages('reshape')
require(reshape)
#

all <- stream_in(file("All.jsonl"))

###GET RID OF NON-TUA ARTEFACTS and focus on one type at a time...
all <- subset(all, TUAfile != "NA")
all <- subset(all, TUAtype == 'Police')


####READ IN TUA TEXT #Thanks, Karthik Ram ###################################
read_file <- function(TUAfile) {
  readChar(TUAfile, file.info(TUAfile)$size)
}

library(dplyr)
all <- all %>%
  rowwise()  %>%
  mutate(tua = read_file(TUAfile))

#tua is the name of the new column

#THEN, convert pros into a legitimate data.frame #Thanks, Daniel Turek
class(all)
#prosdf <- class(as.data.frame(pros))  #not sure I need that
all <- as.data.frame(all)
class(all)

#########################################################################
#########send out for actor dictionary parsing in python

stream_out(all, file("Police3.jsonl"))

##do the stuff in python

#########read back in with all those replacements in S, O, and tua text, and Lemma
all2<- stream_in(file("final_output.jsonl"))

##########convert cells with replacements to thier simple actor name
#write a function for this

all2$S[grepl('protesters', all2$S)] <- 'protesters'
all2$O[grepl('protesters', all2$O)] <- 'protesters'
all2$S[grepl('police', all2$S)] <- 'police'
all2$O[grepl('police', all2$O)] <- 'police'
all2$S[grepl('city', all2$S)] <- 'city'
all2$O[grepl('city', all2$O)] <- 'city'


#CONCATENATE S, V, O, XCOMP to create S_V_O tokens for later inclusion in topic modeling
all2$svo<-paste(all2$S, all2$Lemma, all2$NEGATED, all2$O, all2$XCOMP, sep = "_")

all2$svocon<-paste(all2$S, all2$Lemma, all2$NEGATED, all2$O, all2$XCOMP, all2$svo, sep = " ")

all2$tua_svo<-paste(all2$tua, all2$svo, sep = " ")

#APPEND svocons to Tua text.
all2$tua_svocon<-paste(all2$tua, all2$S, all2$Lemma, all2$NEGATED, all2$O, all2$XCOMP, all2$svo, sep = " ")

######
#View(all$tua_svo[grepl('Bradley Russell', all$tua_svo)])

###################MERGE on city key#################################
```

```
## knowncities.csv is derived from cityvars2.csv with 4 extraneous
## cities removed (moved into unused extracities.csv) and 5 missing
## cities added (see last 5 entries in knowncities.csv)
known <- read.csv("knowncities.csv", header = TRUE, colClasses = c('character', 'character'))
extra <- read.csv("extracities.csv", header = TRUE, colClasses = c('character', 'character'))

known$key <- tolower(gsub('[^[:alpha:]]', '', paste(known$city,known$state)))

rewrite <- function(x) {
 if (x == "cleveland") return ("clevelandoh")
 if (x == "dayton") return ("daytonoh")
 if (x == "jackson") return ("jacksonms")
 if (x == "allentowncity") return ("allentownpa")
 if (x == "lansingcity") return ("lansingmi")
 if (x == "pensacolaflorida") return ("pensacolafl")
 if (x == "cedarfallscedarvalley") return ("cedarfallsia")
 if (x == "everettcity") return ("everettwa")
 if (x == "coachella") return ("coachellavalleyca")
 if (x == "lexingtonfayetteky") return ("lexingtonky")
 if (x == "louisville") return ("louisvillejeffersoncountyky")
 if (x == "newyork") return ("newyorkcityny")
 if (x == "nashville") return ("nashvilledavidsontn")
 string = x
 inlist = known$key
 abbr <- substr(string, 1, nchar(string)-2)
 found <- inlist[grepl(paste("^", string, sep=""), inlist)]
 if (length(found) == 0) {
  found <- inlist[grepl(paste("^", abbr, "$", sep=""), inlist)]
 }
 if (length(found) == 1) {
  return(found)
 } else if (length(found > 1)) {
  warning(paste("found duplicates: ", string))
  return(found)
 } else {
  warning(paste("did not find: ", string))
  return(F)
 }
}

all2$key <- mapply(rewrite, tolower(gsub('[^[:alpha:]]', '', all2$city)))

cityvars4 <- read.csv("cityvars4_policevars.csv", header = TRUE, colClasses = c('character', 'character', 'character', 'character', 'integer',
'character', 'integer', 'numeric', 'numeric', 'numeric', 'character', 'character', 'character', 'numeric', 'numeric', 'numeric', 'numeric',
'numeric', 'numeric', 'factor', 'numeric', 'numeric', 'numeric', 'numeric', 'numeric', 'numeric'), fileEncoding = "UTF-8")
all3 <- merge(all2, cityvars4, by=c("key"), all2.x=TRUE)

#bring in bars from first analyses
polkey <- read.csv("polkey.csv", header = TRUE, colClasses = c('character', 'numeric', 'numeric', 'numeric'), fileEncoding = "UTF-8")
all3 <- merge(all3, polkey, by=c("key"), all.x=TRUE)
head(all3)


#get rid of cities outside our data set
all3 <- subset(all3, key!="lakeworthfl")
all3 <- subset(all3, key!="athensga")
all3 <- subset(all3, key!="lancasterpa")
all3 <- subset(all3, key!="clevelandtn")


##################################################
##################################################


######################
#DATES STUFF
######################

#First, Cleaning
```

```
# Replace "None" with Jan 1, 1999 so it is a date by which you can filter later.
class(all3$article_date)
all3$article_date<-as.character(all3$article_date)


all3clean <- subset(all3, article_date!="")
all3clean <- subset(all3clean, article_date!="+*")
all3clean <- subset(all3clean, article_date!="???")
all3clean <- subset(all3clean, article_date!="none")
all3clean <- subset(all3clean, article_date!="undated")
all3clean <- subset(all3clean, (nchar(article_date) < 9))
all3clean <- subset(all3clean, article_date!="12-12-")  #or was it 12-2


View(unique(all3clean$article_date))

class(all3clean$article_date)
#as.character(all3clean$article_date)

#####################################
#Some FUNCTIONS for DATE EXTRACTION
#####################################
#This function takes a formatted date as Y-m-d and returns weekday
day_from_date <- function(adate = NULL) {
 if (!is.null(adate)) {
   day_list <- c("Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday",
                 "Saturday")
   dow <- try(which(grepl(weekdays(as.Date(adate, "%m-%d-%y")), day_list)), silent=TRUE)
   if(inherits(dow, "try-error")) dow <- NA
   return(dow)
 }
}

# This function takes some text and returns the position on the weekday list
# So return_days("Monday Tuesday Friday")
# will return 2 (for Monday)
return_days <- function(text) {
 if(!is.na(text)){
   if(is.character(text)) {
     day_list <- c("Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday",
                   "Saturday")
     res <- as.numeric(which(sapply(day_list, grepl, text, ignore.case = TRUE)))
     if (length(res) == 0) {
       res <- NA
     } else {
       res<-res[1] # we can add more handling here to deal with multiple weekday mentions
     }
   }
 } else res <- NA
 return(res)
}

# Function takes weekday difference between article_date and text_date and
# calculates approximate date.
day_in_text <- function(article_day, aday, pdate) {
 if(!is.na(pdate)){
   date_diff <- aday - pdate
   if (date_diff > 0) {
     out <- as.Date(article_day, "%m-%d-%y") - date_diff
   } else {
     dd <- ifelse(date_diff != 0, date_diff + 7, 0)
     out <- as.Date(article_day, "%m-%d-%y") - dd
   }
 } else{
   out <- try(as.Date(article_day, "%m-%d-%y") - 1, silent=TRUE)
   if(inherits(out, "try-error")) out <- NA
 }
 return(out)
}
```

```
########clean out weird dates##########
View(unique(all3clean$article_date))
#######################################
# all3clean <- subset(all3clean, article_date!="3-30-12")
# all3clean <- subset(all3clean, article_date!="10-22-13")
# all3clean <- subset(all3clean, article_date!="10-28-13")
# all3clean <- subset(all3clean, article_date!="7-17-12")
# all3clean <- subset(all3clean, article_date!="9-24-12")
# all3clean <- subset(all3clean, article_date!="2-27-12")
# all3clean <- subset(all3clean, article_date!="11-1-13")
# all3clean <- subset(all3clean, article_date!="10-26-13")
# all3clean <- subset(all3clean, article_date!="11-17-12")
# all3clean <- subset(all3clean, article_date!="11-7-10")
# all3clean <- subset(all3clean, article_date!="12-13-1")
# all3clean <- subset(all3clean, article_date!="12-11-12")
all3clean <- subset(all3clean, article_date!="3-23-12")
all3clean <- subset(all3clean, article_date!="10-26-13")
all3clean <- subset(all3clean, article_date!="10-28-13")
all3clean <- subset(all3clean, article_date!="2-1-12")
all3clean <- subset(all3clean, article_date!="4-9-13")
all3clean <- subset(all3clean, article_date!="no date")
all3clean <- subset(all3clean, article_date!="3-30-12")
all3clean <- subset(all3clean, article_date!="11-1-13")
all3clean <- subset(all3clean, article_date!="9-24-12")


###############################################
#NOW, to extract DATES from tua text daynames
###############################################
library(lubridate)
#library(dplyr)
library(readr)

all3clean$article_date<-as.character(all3clean$article_date)

published_day <-  sapply(all3clean$article_date, day_from_date)

text_day <- sapply(all3clean$tua_svo, return_days)
guessing_date_in_text <- mapply(function(a,b,c)
  day_in_text(article_day=a, aday=b, pdate=c),
  a=all3clean$article_date,
  b=published_day,
  c=text_day,
  SIMPLIFY=FALSE)

guessing_date_in_text <- lapply(guessing_date_in_text, function(x) as.character(x))
guessing_date_in_text <- c(do.call(rbind, guessing_date_in_text))
all3clean$text_date <- guessing_date_in_text
all3clean$text_date <- as.Date(all3clean$text_date, "%Y-%m-%d")
View(unique(all3clean$text_date))
#############################
#DAYS running variables
#############################
#days camped
#text_date - campstartdate
class(all3clean$Campaign.Start.Date)
#character
class(all3clean$text_date)
#Date
all3clean$camp_date<-as.Date(all3clean$Campaign.Start.Date, "%m/%d/%y")
all3clean$dayscamped<- all3clean$text_date - all3clean$camp_date
all3clean$dayscamped<-as.numeric(all3clean$dayscamped)
View(unique(all3clean$dayscamped))
View(unique(all3clean$camp_date))
View(unique(all3clean$Campaign.Start.Date))
View(unique(all3clean$text_date))
```

```
#days since zucotti
all3clean$dayszuc<-all3clean$text_date - as.Date("2011-09-17")
all3clean$dayszuc<-as.numeric(all3clean$dayszuc)
View(unique(all3clean$dayszuc))


#####################################################
#####################################################
#REFACTOR POLICE CAPACITY AND CULTURE VARIABLES
#####################################################
#####################################################


library(Hmisc)
all3clean$nonwhite <- as.numeric(cut2(all3clean$nonwhitecops, g=3))
all3clean$budget <- as.numeric(cut2(all3clean$budget, g=3))
all3clean$crime <- as.numeric(cut2(all3clean$crime, g=3))
all3clean$officers <- as.numeric(cut2(all3clean$officers, g=3))
all3clean$cp <- as.numeric(cut2(all3clean$cp, g=3))
#####################################################
#####################################################
#####################################################
#convert from SVO data.frame to TUA data.frame
#####################################################
#####################################################
#####################################################
View(unique(all3clean$tua))

processNicksDF <- function(df) {
  out <- data.frame(tua = character(),
                    text = character(),
                    stability2 = numeric(),
                    pos1 = numeric(),
                    nonwhite = numeric(),
                    budget = numeric(),
                    crime = numeric(),
                    officers = numeric(),
                    cp = numeric (),
                    Obama = numeric(),
                    dayscamped = numeric(),
                    dayszuc = numeric(),
                    Obama_Vote2008 = numeric(),
                    Population = numeric(),
                    pool2 = numeric(),
                    stringsAsFactors = FALSE)
  for(e in unique(df$tua)) {
    tempDF <- df[df$tua==e,]
    prep <- data.frame(tua = e,
                       text = paste(e, paste(tempDF$svocon, collapse=' ')),
                       stability2 = tempDF$stability2[1],
                       pos1 = tempDF$pos1[1],
                       nonwhite = tempDF$nonwhitecops[1],
                       budget = tempDF$budget[1],
                       crime = tempDF$crime[1],
                       officers = tempDF$officers[1],
                       cp = tempDF$cp[1],
                       Obama = tempDF$Obama[1],
                       dayscamped = tempDF$dayscamped[1],
                       dayszuc = tempDF$dayszuc[1],
                       Obama_Vote2008 = tempDF$Obama_Vote2008[1],
                       Population = tempDF$Population[1],
                       pool2 = tempDF$pool2[1],
                       stringsAsFactors = FALSE)
    out <- rbind(out, prep)
  }
  return(out)
}
```

```
#build TUA-based dataframe
df <- processNicksDF(all3clean)



View(df$text)
#clean up svo stuff
df$text <-gsub('NA', '', df$text)
df$text<-gsub('_+', '_', df$text)
df$text<-gsub('"', '', df$text)
df$text<-gsub('-rsb-', '', df$text)
df$text<-gsub('-lsb-', '', df$text)
df$text<-gsub('-lrb-', '', df$text)
df$text<-gsub('-rrb-', '', df$text)
df$text<-gsub(',', '', df$text)
df$text<-gsub(':', '', df$text)
df$text<-gsub('-rsb-', '', df$text)
df$text<-gsub('$', '', df$text)

#reduces some special classes of objects like citynames, days, parks, plazas, and some other named entitites
df$text<-gsub("bo didley community plaza|frank ogawa plaza|calder plaza|cesar chavez plaza|jacome plaza|kanawha plaza|justin
herman plaza|hemming plaza|san jacinto plaza|mckeldin square|thomas square|dewey square|fountain Square", 'xplaza', df$text)
df$text<-gsub("Albany|Albuquerque|Allentown|philly|Anaheim|Anchorage|Ann
Arbor|Arcata|Asheville|Ashtabula|Atlanta|Augusta|Aurora|Austin|Bakersfield|Baltimore|Bangor|Baton
Rouge|Bellingham|Bend|Bethlehem|Binghamton|Birmingham|Bloomington|Boise|Boston|Bowling
Green|Brattleboro|Buffalo|Burlington|Canton|Cedar Falls|Cedar Rapids|Chapel
Hill|Carrboro|Charleston|Charleston|Charlotte|Chattanooga|Chicago|Chico|Cincinnati|Claremont|Clarksville|Cleveland
        |Coachella Valley|Colombus|Colorado Springs|Columbia|Corpus Christi|Dade City|Dallas|Dayton|Daytona Beach|Denver|Des
Moines|Detroit|Dover|Duluth|Easton|El Paso|Eugene|Eureka|Everett|Fairbanks|Fayetteville|Flint|Fort Collins|Fort Lauderdale|Fort
Myers|Fort Wayne|Fort Worth|Frederick|Fresno|Gainesville|Grand
Rapids|Greeley|Greensboro|Harrisburg|Hartford|Honolulu|Houston|Huntington|Indianapolis|Iowa City|Irvine|Ithaca|Jackson|
        Jacksonville|Jersey City|Johnson City|Kalamazoo|Kansas City|Lansing|Las Cruces|Las Vegas|Lawrence|Lexington|Lincoln|Little
Rock|Long Beach|Los Angeles|Louisville|Jefferson
County|Lubbock|Madison|Memphis|Merced|Miami|Milwaukee|Minneapolis|Missoula|Mobile|Mosier|Muncie|Murfreesboro|Muskegon|N
ashville|Davidson|New Haven|New Orleans|New Paltz|New York City|Newark|Norfolk|Norman|Northampton|Oakland|Ogden|Oklahoma
City|Olympia|
        Omaha|Orlando|Palo
Alto|Pensacola|Petaluma|Philadelphia|Phoenix|Pittsburgh|Pocatello|Portland|Portland|Poughkeepsie|Providence|Raleigh|Richmond|Ri
verside|Rochester|Sacramento|Salem|San Antonio|San Diego|San Francisco|San Jose|San Leandro|San Ramon|Santa Ana|Santa
Cruz|Santa Fe|Santa Rosa|Scranton|Seattle|Sebastopol|Sonoma|South Bend|St.
Louis|Stockton|Syracuse|Tacoma|Tallahassee|Tampa|Toledo|Trenton|Tucson|Tulsa|Utica|
        Virginia Beach|Walnut Creek|Washington|West Palm
Beach|Wichita|Wilmington|Wilmington|Worcester|Youngstown|Yuma|Venice", "cityname", df$text, ignore.case=TRUE)
df$text<-gsub("California|tennessee|colorado", 'xstate', df$text,ignore.case=TRUE)
df$text<-gsub("acevedo", 'city', df$text)
df$text<-gsub("cwait", 'peak', df$text)
df$text<-gsub("riot gear", 'riot_gear', df$text)
df$text<-gsub("capt. jeff estes|capt. j.w. estes|steve noblitt|noblitt|pete simpson|pete riot simpson|pettit|andrew pettit|jeff basett|april
skalland|buhr|charles ramsey|gordon ramsey|bobby dodd|capt. doug  weismann|lt. andrew shouse|sgt. paul edwards|lt. rick sucee|jeff
halstead|finnerty|lt. bill|greg mullen|capt. jeff goodwin|chief adam|patrol capt. daryl fisher|sgt. jonathan|sgt. ronnie lance|capt. todd
dykstra|lt. doug mozan|capt. rich stronach|kevin mccormick|marty citynameer|law enforcement|cpl. angelina valuri|troy
thompson|jenna mcculley|lt. kathy flynn|sgt. rich weiner|sgt. steve noblitt|office r iverson|col. jim wolfinbarger|jenna mcculley", 'police',
df$text)
df$text<-gsub("cityname movement|cityname protesters|o'grady|diller|cityname demonstrators|cityname activists|adam platz|george
diller|jose tellez|michael rodriguez|amanda faye mosqueda|johnny okane|charles william florenza|lawrence gregory ziese|reynaldo
crespo|gilbert ceballos|jason brodsky|jason rivera|carl casey|anthony diaz|christopher devcich|moses quiroz|jane one doe|kayla
elizabeth fields|robert jefferson dietrich|occupy cityname|jo jones|rudy sanchez|lauren ross|justin jeffre|occupy baltiprotesters|patrick
robinson|benjamin walden|kathryn heil|nathaniel davis|ryan donald cartwright|erick nutz|chuck nasmith|christina cooke|bobby
donehoo|rob keppler|jonathan bowen|karel sourcre|vanessa maria graber|kerner|james kerner|james r. kerner|sean wildman|benjamin
katz|robbie abalos|alicia dion|kevin flynn|katie christofilis|seth collins", 'protesters', df$text)
df$text<-gsub("cityname deputies|cityname police department|cityname police", 'police', df$text)
df$text<-gsub("bank of america| indep of america|wells fargo|chase bank", 'xbank', df$text)

df$text<-gsub("brooklyn bridge|cityname street bridge", 'xbridge', df$text)
df$text<-gsub("jessica reznieck", 'xprotester', df$text)
df$text<-gsub("saturday|sunday", 'xweekendday', df$text)
df$text<-gsub("monday|tuesday|wednesday|thursday|friday", 'xweekday', df$text)
```

```
df$text<-gsub("grand circus park|academy park|lafayette park|bubier park|thomas square park|monroepark|veinte de agosto|smith park|westlake park|zucotti park|cityname-jefferson park|woodruff park|oak cliff park|pioneer park|garfield park|piatt park", 'xpark', df$text)
df$text<-gsub("cielo vista mall|valley malls", 'xshop', df$text)

#fix some mistakes from prior substitutions
df$text<-gsub('protestersers', 'protesters', df$text)
df$text<-gsub('protestersed', 'protested', df$text)
df$text<-gsub('protestersing', 'protesting', df$text)
df$text<-gsub('protesterors', 'protesters', df$text)
df$text<-gsub('protestors', 'protesters', df$text)
df$text<-gsub('police police police', 'police police', df$text)
df$text<-gsub('protesters protesters protesters', 'protesters protesters', df$text)
df$text<-gsub('police police police', 'police police', df$text)
df$text<-gsub('protesters protesters protesters', 'protesters protesters', df$text)
df$text<-gsub('police police police', 'police police', df$text)
df$text<-gsub('protesters protesters protesters', 'protesters protesters', df$text)
df$text<-gsub('police police', 'police', df$text)
df$text<-gsub('protesters protesters', 'protesters', df$text)

#make all parks and plazas the same thing
df$text<-gsub('xpark|xplaza|park','xpark', df$text)



View(df$text)

# make stability easier to work with
# df$stability2<-ceiling(df$stability *3/4-1)
# table(df$stability2)

#remove all rows with missing data
df <- na.omit(df)

save(df,file="DissTUAstm_police6.Rda")

##############################################################################################################
##########################################################
##############################################################################################################
##########################################################
##############################################################################################################
##########################################################
#If you want to merge visualizations from protester models, be sure to first rename all those objects
##############################################################################################################
##########################################################
prodf<-df
prodocs<-docs
proout<-out
provoab<-vocab
prometa<-meta
proprocessed<-processed
##############################################################################################################
##########################################################
##############################################################################################################
##########################################################
##############################################################################################################
##########################################################
##############################################################################################################
##########################################################


load("DissTUAstm_police6.Rda")


# #with population binary
# df$pool3 <- as.numeric(cut2(df$pool2, g=2))
# table(df$pool3)
# save(df,file="DissTUAstm_Wednesday.Rda")
```

```
################################################
#STRUCTURAL TOPIC MODELING -
################################################

#install.packages('stm')
library(stm)
install.packages('igraph')
library(igraph)
remove.packages('igraph')

################################################
#PREPROCESSING
################################################

#stemming/stopword removal, etc. also duplicates all data for a metadata file created in next step
?textProcessor
processed <- textProcessor(df$text, metadata=df, stem=FALSE, removepunctuation=FALSE)

#structure and index for usage in the stm model. Verify no-missingness. can remove low frequency words using 'lower.thresh' option.
See ?prepDocuments for more info
?prepDocuments
out <- prepDocuments(processed$documents, processed$vocab, processed$meta, lower.thresh = 0)


#CREATES docs and (identical) meta dataframes and a vocab vector of all words in the corpus -- these are the elements of stm
docs <- out$documents
vocab <- out$vocab
meta <-out$meta

################################################
#RUN AND CHOOSE THE BEST TOPIC MODEL
################################################


police15Selectx <- selectModel(out$documents,out$vocab,K=15,prevalence =~ pos1 + pool2 + stability2 + s(dayscamped) +
stability2*dayscamped + nonwhite + budget + crime + officers + cp, max.em.its=275, data=out$meta, runs=10)
x4<- stm(out$documents,out$vocab,K=15,prevalence =~ pos1 + pool2 + stability2 + s(dayscamped) + stability2*dayscamped + nonwhite
+ budget + crime + officers + cp, max.em.its=275, data=out$meta, seed=1867757)
save(x4, file="stmpolice3.Rda")


################################################
#BEGIN INTERPRETING THE MODELS -- FUN!!!
################################################

labelTopics(x4, topics=c(1:15), n=15)

x4$settings$seed
1867757

# ###Read DOCUMENTS that are highly correlated with the topics you specify using findThoughts() function
# #object 'thoughts1' contains 2 documents about topic 1. 'texts=shortdoc,' gives you just the first 250 words
thoughts1<-findThoughts(x4, texts=df$text, n=2, topics=2)$docs[[1]]
# #will show you the output
?findThoughts
plotQuote(thoughts1, width=250, main="Topic")
?plotQuote

##see PROPORTION OF EACH TOPIC in the entire CORPUS. Just insert your STM output
plot.STM(x4, type="summary", main = " ", topics= c(1:15), xlim=c(0,.4))



#########################################################################################################
########################
# STM: SEE HOW PREVALENCE OF TOPICS VARIES ACROSS DOCUMENTS ACCORDING TO DOCUMENT COVARIATES (METADATA
#########################################################################################################
########################
```

```
pdf.f <- function(f, file, ...) {
  cat(sprintf('Writing %s\n', file))
  pdf(file, ...)
  on.exit(dev.off())
  f()
}


####################################
####################################
pol15xpool2 <- stm(out$documents,out$vocab,K=15,prevalence =~ pos1 + stability2 + pool2*s(dayscamped) + nonwhite + budget +
crime + officers + cp, max.em.its=250, data=out$meta, seed=1867757)

polpool2 <- estimateEffect(c(4, 15, 12, 9, 7, 8, 10) ~ pos1 + stability2 + cp + nonwhite + budget + crime + officers + pool2*s(dayscamped),
pol15xpool2,
                   metadata=meta, uncertainty="None")
#################
save(pol15xpool2, file="pol15xpool2.Rda")
save(polpool2, file="polpool2.Rda")
################
#pool2 targeting individuals
#
#

pr <- function(){
  layout(matrix(1:3, ncol=1, byrow=TRUE))
  par(oma=c(4,6,2,1), mar=c(1,1,1,1), mgp=c(2,1,0),
     cex.axis=1.5)
  plot.estimateEffect(polpool2, covariate="dayscamped", model=pol15xpool2,topic=c(4),
                method="continuous", moderator="pool2", xaxt='n', xlim=c(-60,166), moderator.value=1, add=FALSE, ylim=c(-.3,.5),
ylab="Expected Topic Proportion",
                linecol='red', printlegend=FALSE)
  plot.estimateEffect(polpool2, covariate="dayscamped", model=pol15xpool2,topic=c(4),
                method="continuous", moderator="pool2", xaxt='n',moderator.value=c(3), add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
  plot.estimateEffect(polpool2, covariate="dayscamped", model=pol15xpool2,topic=c(4),
                method="continuous", moderator="pool2", moderator.value=c(5), add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
  legend("top", c('Smaller, Less Liberal Population', 'Modal Population', 'Large, Liberal Population'),
       lwd=2, col=c('red', 'blue', 'darkgreen'), bty="n")

  #Topic 19 "Camp Activities"
  plot.estimateEffect(polpool2, covariate="dayscamped", model=pol15xpool2,topic=c(15),
                method="continuous", moderator="pool2", moderator.value=1, add=FALSE, ylim=c(-.3,.5), ylab="Expected Topic Proportion",
xlab="Days since local camp established",
                xlim=c(-60,166), xaxt='n', linecol='red', printlegend=FALSE)
  plot.estimateEffect(polpool2, covariate="dayscamped", model=pol15xpool2,topic=c(15),
                method="continuous", moderator="pool2", moderator.value=c(3), add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
  plot.estimateEffect(polpool2, covariate="dayscamped", model=pol15xpool2,topic=c(15),
                method="continuous", moderator="pool2", moderator.value=c(5), add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
  mtext("Expected Topic Proportion", 2, line=4)

  plot.estimateEffect(polpool2, covariate="dayscamped", model=pol15xpool2,topic=c(12),
                method="continuous", moderator="pool2", moderator.value=1, add=FALSE, ylim=c(-.3,.4), ylab="Expected Topic Proportion",
xlab="Days since local camp established",
                xlim=c(-60,166), linecol='red', printlegend=FALSE)
  plot.estimateEffect(polpool2, covariate="dayscamped", model=pol15xpool2,topic=c(12),
                method="continuous", moderator="pool2", moderator.value=c(3), add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
  plot.estimateEffect(polpool2, covariate="dayscamped", model=pol15xpool2,topic=c(12),
                method="continuous", moderator="pool2", moderator.value=c(5), add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
  mtext("Days Since Local Camp Established", 1, line=3)

}
```

```
pdf.f(pr, file= '/Users/nickbadams/Dropbox/NickDiss/Chapter 6 Figures/intpool_ind_w3.pdf',
    width=4, height=7)
###################
####################
###################
####################
#activities pool2 targeting camps

pt  <- function(){
 layout(matrix(1:3, ncol=1, byrow=TRUE))
 par(oma=c(4,6,2,1), mar=c(1,1,1,1), mgp=c(2,1,0),
    cex.axis=1.5)
 plot.estimateEffect(polpool2, covariate="dayscamped", model=pol15xpool2,topic=c(9),
                method="continuous", moderator="pool2", xaxt='n', xlim=c(-60,166), moderator.value=1, add=FALSE, ylim=c(-.3,.5),
ylab="Expected Topic Proportion",
                linecol='red', printlegend=FALSE)
plot.estimateEffect(polpool2, covariate="dayscamped", model=pol15xpool2,topic=c(9),
            method="continuous", moderator="pool2", xaxt='n',moderator.value=c(3), add='TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
plot.estimateEffect(polpool2, covariate="dayscamped", model=pol15xpool2,topic=c(9),
            method="continuous", moderator="pool2", moderator.value=c(5), add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
legend("top", c('Smaller, Less Liberal Population', 'Modal Population', 'Large, Liberal Population'),
     lwd=2, col=c('red', 'blue', 'darkgreen'), bty="n")

#Topic 19 "Camp Activities"
plot.estimateEffect(polpool2, covariate="dayscamped", model=pol15xpool2,topic=c(7),
            method="continuous", moderator="pool2", moderator.value=1, add=FALSE, ylim=c(-.3,.5), ylab="Expected Topic Proportion",
xlab="Days since local camp established",
            xlim=c(-60,166), xaxt='n', linecol='red', printlegend=FALSE)
plot.estimateEffect(polpool2, covariate="dayscamped", model=pol15xpool2,topic=c(7),
            method="continuous", moderator="pool2", moderator.value=c(3), add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
plot.estimateEffect(polpool2, covariate="dayscamped", model=pol15xpool2,topic=c(7),
            method="continuous", moderator="pool2", moderator.value=c(5), add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
mtext("Expected Topic Proportion", 2, line=4)

plot.estimateEffect(polpool2, covariate="dayscamped", model=pol15xpool2,topic=c(8),
            method="continuous", moderator="pool2", moderator.value=1, add=FALSE, ylim=c(-.3,.4), ylab="Expected Topic Proportion",
xlab="Days since local camp established",
            xlim=c(-60,166), linecol='red', printlegend=FALSE)
plot.estimateEffect(polpool2, covariate="dayscamped", model=pol15xpool2,topic=c(8),
            method="continuous", moderator="pool2", moderator.value=c(3), add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
plot.estimateEffect(polpool2, covariate="dayscamped", model=pol15xpool2,topic=c(8),
            method="continuous", moderator="pool2", moderator.value=c(5), add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
mtext("Days Since Local Camp Established", 1, line=3)

}
pdf.f(pt, file= '/Users/nickbadams/Dropbox/NickDiss/Chapter 6 Figures/intpool_camp_w3.pdf',
    width=4, height=7)



###################
####################
###################
####################################
####################################


####################################
####################################

pol15xstability2 <- stm(out$documents,out$vocab,K=15,prevalence =~ pos1 + stability2*s(dayscamped) + pool2 + nonwhite + budget +
crime + officers + cp, max.em.its=250, data=out$meta, seed=1867757)
```

```
#pol15xpool2 <- stm(out$documents,out$vocab,K=15,prevalence =~ pos1 + stability2 + pool2*s(dayscamped) + nonwhite + budget +
crime + officers + cp, max.em.its=250, data=out$meta, seed=1867757)
polstability2 <- estimateEffect(c(4, 15, 12, 9, 7, 8, 10) ~ pos1 + stability2*s(dayscamped) + cp + nonwhite + budget + crime + officers +
pool2, pol15xstability2,
                metadata=meta, uncertainty="None")
#################

#               ############        STABILITY INTERACTIONS    ##############################################
#################


u <- function(){
 layout(matrix(1:3, ncol=1, byrow=TRUE))
 par(oma=c(4,6,2,1), mar=c(1,1,1,1), mgp=c(2,1,0),
   cex.axis=1.5)
 plot.estimateEffect(polstability2, covariate="dayscamped", model=pol15xstability2,topic=c(4),
             method="continuous", moderator="stability2", xaxt='n', xlim=c(-60,166), moderator.value=1, add=FALSE, ylim=c(-.3,.5),
ylab="Expected Topic Proportion",
             linecol='red', printlegend=FALSE)
 plot.estimateEffect(polstability2, covariate="dayscamped", model=pol15xstability2,topic=c(4),
             method="continuous", moderator="stability2", xaxt='n',moderator.value=2, add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(polstability2, covariate="dayscamped", model=pol15xstability2,topic=c(4),
             method="continuous", moderator="stability2", moderator.value=3, add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 legend("top", c('Stable', 'Less Stable', 'Least Stable'),
      lwd=2, col=c('red', 'blue', 'darkgreen'), bty="n")

 #Topic 19 "Camp Activities"
 plot.estimateEffect(polstability2, covariate="dayscamped", model=pol15xstability2,topic=c(15),
             method="continuous", moderator="stability2", moderator.value=1, add=FALSE, ylim=c(-.3,.5), ylab="Expected Topic
Proportion", xlab="Days since local camp established",
             xlim=c(-60,166), xaxt='n', linecol='red', printlegend=FALSE)
 plot.estimateEffect(polstability2, covariate="dayscamped", model=pol15xstability2,topic=c(15),
             method="continuous", moderator="stability2", moderator.value=2, add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(polstability2, covariate="dayscamped", model=pol15xstability2,topic=c(15),
             method="continuous", moderator="stability2", moderator.value=3, add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 mtext("Expected Topic Proportion", 2, line=4)


 plot.estimateEffect(polstability2, covariate="dayscamped", model=pol15xstability2,topic=c(12),
             method="continuous", moderator="stability2", moderator.value=1, add=FALSE, ylim=c(-.3,.4), ylab="Expected Topic
Proportion", xlab="Days since local camp established",
             xlim=c(-60,166), linecol='red', printlegend=FALSE)
 plot.estimateEffect(polstability2, covariate="dayscamped", model=pol15xstability2,topic=c(12),
             method="continuous", moderator="stability2", moderator.value=2, add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(polstability2, covariate="dayscamped", model=pol15xstability2,topic=c(12),
             method="continuous", moderator="stability2", moderator.value=3, add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 mtext("Days Since Local Camp Established", 1, line=3)


}
pdf.f(u, file= '/Users/nickbadams/Dropbox/NickDiss/Chapter 6 Figures/intstability_individuals.pdf',
    width=4, height=7)


v <- function(){
 layout(matrix(1:3, ncol=1, byrow=TRUE))
 par(oma=c(4,6,2,1), mar=c(1,1,1,1), mgp=c(2,1,0),
   cex.axis=1.5)
 plot.estimateEffect(polstability2, covariate="dayscamped", model=pol15xstability2,topic=c(9),
             method="continuous", moderator="stability2", xaxt='n', xlim=c(-60,166), moderator.value=1, add=FALSE, ylim=c(-.3,.5),
ylab="Expected Topic Proportion",
             linecol='red', printlegend=FALSE)
 plot.estimateEffect(polstability2, covariate="dayscamped", model=pol15xstability2,topic=c(9),
```

```
            method="continuous", moderator="stability2", xaxt='n',moderator.value=2, add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(polstability2, covariate="dayscamped", model=pol15xstability2,topic=c(9),
            method="continuous", moderator="stability2", moderator.value=3, add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 legend("top", c('Stable', 'Less Stable', 'Least Stable'),
      lwd=2, col=c('red', 'blue', 'darkgreen'), bty="n")

 #Topic 19 "Camp Activities"
 plot.estimateEffect(polstability2, covariate="dayscamped", model=pol15xstability2,topic=c(7),
            method="continuous", moderator="stability2", moderator.value=1, add=FALSE, ylim=c(-.3,.5), ylab="Expected Topic
Proportion", xlab="Days since local camp established",
            xlim=c(-60,166), xaxt='n', linecol='red', printlegend=FALSE)
 plot.estimateEffect(polstability2, covariate="dayscamped", model=pol15xstability2,topic=c(7),
            method="continuous", moderator="stability2", moderator.value=2, add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(polstability2, covariate="dayscamped", model=pol15xstability2,topic=c(7),
            method="continuous", moderator="stability2", moderator.value=3, add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 mtext("Expected Topic Proportion", 2, line=4)


 plot.estimateEffect(polstability2, covariate="dayscamped", model=pol15xstability2,topic=c(8),
            method="continuous", moderator="stability2", moderator.value=1, add=FALSE, ylim=c(-.3,.4), ylab="Expected Topic
Proportion", xlab="Days since local camp established",
            xlim=c(-60,166), linecol='red', printlegend=FALSE)
 plot.estimateEffect(polstability2, covariate="dayscamped", model=pol15xstability2,topic=c(8),
            method="continuous", moderator="stability2", moderator.value=2, add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(polstability2, covariate="dayscamped", model=pol15xstability2,topic=c(8),
            method="continuous", moderator="stability2", moderator.value=3, add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 mtext("Days Since Local Camp Established", 1, line=3)


}
pdf.f(v, file= '/Users/nickbadams/Dropbox/NickDiss/Chapter 6 Figures/intstability_camp.pdf',
    width=4, height=7)
####################
####################
#####################################

#################
#                ###########       GOV TYPE INTERACTIONS    ###########################################
################
####################################
####################################
pol15xpos1 <- stm(out$documents,out$vocab,K=15,prevalence =~ pos1*s(dayscamped) + stability2 + pool2 + nonwhite + budget +
crime + officers + cp, max.em.its=250, data=out$meta, seed=1867757)

####################################
####################################

polpos1 <- estimateEffect(c(4, 15, 12, 9, 7, 8, 10) ~ pos1*s(dayscamped) + stability2 + cp + nonwhite + budget + crime + officers + pool2,
pol15xpos1,
                    metadata=meta, uncertainty="None")

w <- function(){
 layout(matrix(1:3, ncol=1, byrow=TRUE))
 par(oma=c(4,6,2,1), mar=c(1,1,1,1), mgp=c(2,1,0),
    cex.axis=1.5)
 plot.estimateEffect(polpos1, covariate="dayscamped", model=pol15xpos1,topic=c(4),
            method="continuous", moderator="pos1", xaxt='n', xlim=c(-60,166), moderator.value=1, add=FALSE, ylim=c(-.3,.5),
ylab="Expected Topic Proportion",
            linecol='red', printlegend=FALSE)
 plot.estimateEffect(polpos1, covariate="dayscamped", model=pol15xpos1,topic=c(4),
            method="continuous", moderator="pos1", xaxt='n',moderator.value=2, add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(polpos1, covariate="dayscamped", model=pol15xpos1,topic=c(4),
```

```
            method="continuous", moderator="pos1", moderator.value=3, add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 legend("top", c('Fewest Centers of Power', 'More Centers of Power', 'Most Centers of Power'),
      lwd=2, col=c('red', 'blue', 'darkgreen'), bty="n")

 #Topic 19 "Camp Activities"
 plot.estimateEffect(polpos1, covariate="dayscamped", model=pol15xpos1,topic=c(15),
            method="continuous", moderator="pos1", moderator.value=1, add=FALSE, ylim=c(-.3,.5), ylab="Expected Topic Proportion",
xlab="Days since local camp established",
            xlim=c(-60,166), xaxt='n', linecol='red', printlegend=FALSE)
 plot.estimateEffect(polpos1, covariate="dayscamped", model=pol15xpos1,topic=c(15),
            method="continuous", moderator="pos1", moderator.value=2, add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(polpos1, covariate="dayscamped", model=pol15xpos1,topic=c(15),
            method="continuous", moderator="pos1", moderator.value=3, add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 mtext("Expected Topic Proportion", 2, line=4)

 plot.estimateEffect(polpos1, covariate="dayscamped", model=pol15xpos1,topic=c(12),
            method="continuous", moderator="pos1", moderator.value=1, add=FALSE, ylim=c(-.3,.4), ylab="Expected Topic Proportion",
xlab="Days since local camp established",
            xlim=c(-60,166), linecol='red', printlegend=FALSE)
 plot.estimateEffect(polpos1, covariate="dayscamped", model=pol15xpos1,topic=c(12),
            method="continuous", moderator="pos1", moderator.value=2, add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(polpos1, covariate="dayscamped", model=pol15xpos1,topic=c(12),
            method="continuous", moderator="pos1", moderator.value=3, add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 mtext("Days Since Local Camp Established", 1, line=3)
}

pdf.f(w, file= '/Users/nickbadams/Dropbox/NickDiss/Chapter 6 Figures/intPos1_inds.pdf',
    width=4, height=7)
####################
####################
####################
####################

y <- function(){
 layout(matrix(1:3, ncol=1, byrow=TRUE))
 par(oma=c(4,6,2,1), mar=c(1,1,1,1), mgp=c(2,1,0),
    cex.axis=1.5)
 plot.estimateEffect(polpos1, covariate="dayscamped", model=pol15xpos1,topic=c(9),
            method="continuous", moderator="pos1", xaxt='n', xlim=c(-60,166), moderator.value=1, add=FALSE, ylim=c(-.3,.5),
ylab="Expected Topic Proportion",
            linecol='red', printlegend=FALSE)
 plot.estimateEffect(polpos1, covariate="dayscamped", model=pol15xpos1,topic=c(9),
            method="continuous", moderator="pos1", xaxt='n',moderator.value=2, add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(polpos1, covariate="dayscamped", model=pol15xpos1,topic=c(9),
            method="continuous", moderator="pos1", moderator.value=3, add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 legend("top", c('Fewest Centers of Power', 'More Centers of Power', 'Most Centers of Power'),
      lwd=2, col=c('red', 'blue', 'darkgreen'), bty="n")

 #Topic 19 "Camp Activities"
 plot.estimateEffect(polpos1, covariate="dayscamped", model=pol15xpos1,topic=c(7),
            method="continuous", moderator="pos1", moderator.value=1, add=FALSE, ylim=c(-.3,.5), ylab="Expected Topic Proportion",
xlab="Days since local camp established",
            xlim=c(-60,166), xaxt='n', linecol='red', printlegend=FALSE)
 plot.estimateEffect(polpos1, covariate="dayscamped", model=pol15xpos1,topic=c(7),
            method="continuous", moderator="pos1", moderator.value=2, add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(polpos1, covariate="dayscamped", model=pol15xpos1,topic=c(7),
            method="continuous", moderator="pos1", moderator.value=3, add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 mtext("Expected Topic Proportion", 2, line=4)

 plot.estimateEffect(polpos1, covariate="dayscamped", model=pol15xpos1,topic=c(8),
```

```
            method="continuous", moderator="pos1", moderator.value=1, add=FALSE, ylim=c(-.3,.4), ylab="Expected Topic Proportion",
xlab="Days since local camp established",
            xlim=c(-60,166), linecol='red', printlegend=FALSE)
 plot.estimateEffect(polpos1, covariate="dayscamped", model=pol15xpos1,topic=c(8),
            method="continuous", moderator="pos1", moderator.value=2, add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(polpos1, covariate="dayscamped", model=pol15xpos1,topic=c(8),
            method="continuous", moderator="pos1", moderator.value=3, add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 mtext("Days Since Local Camp Established", 1, line=3)

}
pdf.f(y, file= '/Users/nickbadams/Dropbox/NickDiss/Chapter 6 Figures/intPos1_camp.pdf',
    width=4, height=7)

###################
###################
###################
###################

pol15xbudget <- stm(out$documents,out$vocab,K=15,prevalence =~ pos1 + stability2 + pool2 + nonwhite + budget*s(dayscamped) +
crime + officers + cp, max.em.its=250, data=out$meta, seed=1867757)

####################################
####################################

polbudget <- estimateEffect(c(4, 15, 12, 9, 7, 8, 10) ~ pos1 + stability2 + cp + nonwhite + budget*s(dayscamped) + crime + officers +
pool2, pol15xbudget,
              metadata=meta, uncertainty="None")



z <- function(){
 layout(matrix(1:3, ncol=1, byrow=TRUE))
 par(oma=c(4,6,2,1), mar=c(1,1,1,1), mgp=c(2,1,0),
   cex.axis=1.5)
 plot.estimateEffect(polbudget, covariate="dayscamped", model=pol15xbudget,topic=c(4),
            method="continuous", moderator="budget", xaxt='n', xlim=c(-60,166), moderator.value=1, add=FALSE, ylim=c(-.3,.5),
ylab="Expected Topic Proportion",
            linecol='red', printlegend=FALSE)
 plot.estimateEffect(polbudget, covariate="dayscamped", model=pol15xbudget,topic=c(4),
            method="continuous", moderator="budget", xaxt='n',moderator.value=2, add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(polbudget, covariate="dayscamped", model=pol15xbudget,topic=c(4),
            method="continuous", moderator="budget", moderator.value=3, add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 legend("top", c('Lowest Budget per capita', 'Modal Budget', 'Highest Budget'),
      lwd=2, col=c('red', 'blue', 'darkgreen'), bty="n")

 plot.estimateEffect(polbudget, covariate="dayscamped", model=pol15xbudget,topic=c(15),
            method="continuous", moderator="budget", moderator.value=1, add=FALSE, ylim=c(-.3,.5), ylab="Expected Topic
Proportion", xlab="Days since local camp established",
            xlim=c(-60,166), xaxt='n', linecol='red', printlegend=FALSE)
 plot.estimateEffect(polbudget, covariate="dayscamped", model=pol15xbudget,topic=c(15),
            method="continuous", moderator="budget", moderator.value=2, add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(polbudget, covariate="dayscamped", model=pol15xbudget,topic=c(15),
            method="continuous", moderator="budget", moderator.value=3, add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 mtext("Expected Topic Proportion", 2, line=4)

 plot.estimateEffect(polbudget, covariate="dayscamped", model=pol15xbudget,topic=c(12),
            method="continuous", moderator="budget", moderator.value=1, add=FALSE, ylim=c(-.3,.4), ylab="Expected Topic
Proportion", xlab="Days since local camp established",
            xlim=c(-60,166), linecol='red', printlegend=FALSE)
 plot.estimateEffect(polbudget, covariate="dayscamped", model=pol15xbudget,topic=c(12),
            method="continuous", moderator="budget", moderator.value=2, add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(polbudget, covariate="dayscamped", model=pol15xbudget,topic=c(12),
```

```
            method="continuous", moderator="budget", moderator.value=3, add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 mtext("Days Since Local Camp Established", 1, line=3)

}
pdf.f(z, file= '/Users/nickbadams/Dropbox/NickDiss/Chapter 6 Figures/intBudget_inds.pdf',
    width=4, height=7)

###################
###################
###################
###################
###################
###################
###################
aa <- function(){
 layout(matrix(1:3, ncol=1, byrow=TRUE))
 par(oma=c(4,6,2,1), mar=c(1,1,1,1), mgp=c(2,1,0),
    cex.axis=1.5)
 plot.estimateEffect(polbudget, covariate="dayscamped", model=pol15xbudget,topic=c(9),
            method="continuous", moderator="budget", xaxt='n', xlim=c(-60,166), moderator.value=1, add=FALSE, ylim=c(-.3,.5),
ylab="Expected Topic Proportion",
            linecol='red', printlegend=FALSE)
 plot.estimateEffect(polbudget, covariate="dayscamped", model=pol15xbudget,topic=c(9),
            method="continuous", moderator="budget", xaxt='n',moderator.value=2, add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(polbudget, covariate="dayscamped", model=pol15xbudget,topic=c(9),
            method="continuous", moderator="budget", moderator.value=3, add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 legend("top", c('Lowest Budget per capita', 'Modal Budget', 'Highest Budget'),
       lwd=2, col=c('red', 'blue', 'darkgreen'), bty="n")

 plot.estimateEffect(polbudget, covariate="dayscamped", model=pol15xbudget,topic=c(7),
            method="continuous", moderator="budget", moderator.value=1, add=FALSE, ylim=c(-.3,.5), ylab="Expected Topic
Proportion", xlab="Days since local camp established",
            xlim=c(-60,166), xaxt='n', linecol='red', printlegend=FALSE)
 plot.estimateEffect(polbudget, covariate="dayscamped", model=pol15xbudget,topic=c(7),
            method="continuous", moderator="budget", moderator.value=2, add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(polbudget, covariate="dayscamped", model=pol15xbudget,topic=c(7),
            method="continuous", moderator="budget", moderator.value=3, add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 mtext("Expected Topic Proportion", 2, line=4)

 plot.estimateEffect(polbudget, covariate="dayscamped", model=pol15xbudget,topic=c(8),
            method="continuous", moderator="budget", moderator.value=1, add=FALSE, ylim=c(-.3,.4), ylab="Expected Topic
Proportion", xlab="Days since local camp established",
            xlim=c(-60,166), linecol='red', printlegend=FALSE)
 plot.estimateEffect(polbudget, covariate="dayscamped", model=pol15xbudget,topic=c(8),
            method="continuous", moderator="budget", moderator.value=2, add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(polbudget, covariate="dayscamped", model=pol15xbudget,topic=c(8),
            method="continuous", moderator="budget", moderator.value=3, add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 mtext("Days Since Local Camp Established", 1, line=3)

}
pdf.f(aa, file= '/Users/nickbadams/Dropbox/NickDiss/Chapter 6 Figures/intBudgetCamps.pdf',
    width=4, height=7)



###################################
###################################
###################################
###################################
###################################
pol15xofficers <- stm(out$documents,out$vocab,K=15,prevalence =~ pos1 + stability2 + pool2 + nonwhite + budget + crime +
officers*s(dayscamped) + cp, max.em.its=250, data=out$meta, seed=1867757)
```

```
####################################
####################################

polofficers <- estimateEffect(c(4, 15, 12, 9, 7, 8, 10) ~ pos1 + stability2 + cp + nonwhite + budget + crime + officers*s(dayscamped) +
pool2, pol15xofficers,
                    metadata=meta, uncertainty="None")

bb <- function(){
 layout(matrix(1:3, ncol=1, byrow=TRUE))
 par(oma=c(4,6,2,1), mar=c(1,1,1,1), mgp=c(2,1,0),
    cex.axis=1.5)
 plot.estimateEffect(polofficers, covariate="dayscamped", model=pol15xofficers,topic=c(4),
                method="continuous", moderator="officers", xaxt='n', xlim=c(-60,166), moderator.value=1, add=FALSE, ylim=c(-.3,.5),
ylab="Expected Topic Proportion",
                linecol='red', printlegend=FALSE)
 plot.estimateEffect(polofficers, covariate="dayscamped", model=pol15xofficers,topic=c(4),
                method="continuous", moderator="officers", xaxt='n',moderator.value=2, add='TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(polofficers, covariate="dayscamped", model=pol15xofficers,topic=c(4),
                method="continuous", moderator="officers", moderator.value=3, add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 legend("top", c('Fewest Officers per capita', 'Modal Officers', 'Most Officers'),
       lwd=2, col=c('red', 'blue', 'darkgreen'), bty="n")

 plot.estimateEffect(polofficers, covariate="dayscamped", model=pol15xofficers,topic=c(15),
                method="continuous", moderator="officers", moderator.value=1, add=FALSE, ylim=c(-.3,.5), ylab="Expected Topic
Proportion", xlab="Days since local camp established",
                xlim=c(-60,166), xaxt='n', linecol='red', printlegend=FALSE)
 plot.estimateEffect(polofficers, covariate="dayscamped", model=pol15xofficers,topic=c(15),
                method="continuous", moderator="officers", moderator.value=2, add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(polofficers, covariate="dayscamped", model=pol15xofficers,topic=c(15),
                method="continuous", moderator="officers", moderator.value=3, add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 mtext("Expected Topic Proportion", 2, line=4)

 plot.estimateEffect(polofficers, covariate="dayscamped", model=pol15xofficers,topic=c(12),
                method="continuous", moderator="officers", moderator.value=1, add=FALSE, ylim=c(-.3,.4), ylab="Expected Topic
Proportion", xlab="Days since local camp established",
                xlim=c(-60,166), linecol='red', printlegend=FALSE)
 plot.estimateEffect(polofficers, covariate="dayscamped", model=pol15xofficers,topic=c(12),
                method="continuous", moderator="officers", moderator.value=2, add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(polofficers, covariate="dayscamped", model=pol15xofficers,topic=c(12),
                method="continuous", moderator="officers", moderator.value=3, add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 mtext("Days Since Local Camp Established", 1, line=3)

}
pdf.f(bb, file= '/Users/nickbadams/Dropbox/NickDiss/Chapter 6 Figures/intOfficersInds.pdf',
    width=4, height=7)


cc <- function(){
 layout(matrix(1:3, ncol=1, byrow=TRUE))
 par(oma=c(4,6,2,1), mar=c(1,1,1,1), mgp=c(2,1,0),
    cex.axis=1.5)
 plot.estimateEffect(polofficers, covariate="dayscamped", model=pol15xofficers,topic=c(9),
                method="continuous", moderator="officers", xaxt='n', xlim=c(-60,166), moderator.value=1, add=FALSE, ylim=c(-.3,.5),
ylab="Expected Topic Proportion",
                linecol='red', printlegend=FALSE)
 plot.estimateEffect(polofficers, covariate="dayscamped", model=pol15xofficers,topic=c(9),
                method="continuous", moderator="officers", xaxt='n',moderator.value=2, add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(polofficers, covariate="dayscamped", model=pol15xofficers,topic=c(9),
                method="continuous", moderator="officers", moderator.value=3, add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 legend("top", c('Fewest Officers per capita', 'Modal Officers', 'Most Officers'),
```

```
      lwd=2, col=c('red', 'blue', 'darkgreen'), bty="n")

 plot.estimateEffect(polofficers, covariate="dayscamped", model=pol15xofficers,topic=c(7),
              method="continuous", moderator="officers", moderator.value=1, add=FALSE, ylim=c(-.3,.5), ylab="Expected Topic
Proportion", xlab="Days since local camp established",
              xlim=c(-60,166), xaxt='n', linecol='red', printlegend=FALSE)
 plot.estimateEffect(polofficers, covariate="dayscamped", model=pol15xofficers,topic=c(7),
              method="continuous", moderator="officers", moderator.value=2, add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(polofficers, covariate="dayscamped", model=pol15xofficers,topic=c(7),
              method="continuous", moderator="officers", moderator.value=3, add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 mtext("Expected Topic Proportion", 2, line=4)

 plot.estimateEffect(polofficers, covariate="dayscamped", model=pol15xofficers,topic=c(8),
              method="continuous", moderator="officers", moderator.value=1, add=FALSE, ylim=c(-.3,.4), ylab="Expected Topic
Proportion", xlab="Days since local camp established",
              xlim=c(-60,166), linecol='red', printlegend=FALSE)
 plot.estimateEffect(polofficers, covariate="dayscamped", model=pol15xofficers,topic=c(8),
              method="continuous", moderator="officers", moderator.value=2, add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(polofficers, covariate="dayscamped", model=pol15xofficers,topic=c(8),
              method="continuous", moderator="officers", moderator.value=3, add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 mtext("Days Since Local Camp Established", 1, line=3)

}
pdf.f(cc, file= '/Users/nickbadams/Dropbox/NickDiss/Chapter 6 Figures/intOfficersCamps.pdf',
    width=4, height=7)




####################################
####################################
####################################
####################################
####################################
pol15xcp <- stm(out$documents,out$vocab,K=15,prevalence =~ pos1 + stability2 + pool2 + nonwhite + budget + crime + officers +
cp*s(dayscamped), max.em.its=250, data=out$meta, seed=1867757)

####################################
####################################

polcp <- estimateEffect(c(4, 15, 12, 9, 7, 8, 10) ~ pos1 + stability2 + cp*s(dayscamped) + nonwhite + budget + crime + officers + pool2,
pol15xcp,
                    metadata=meta, uncertainty="None")

dd <- function(){
 layout(matrix(1:3, ncol=1, byrow=TRUE))
 par(oma=c(4,6,2,1), mar=c(1,1,1,1), mgp=c(2,1,0),
   cex.axis=1.5)
 plot.estimateEffect(polcp, covariate="dayscamped", model=pol15cp,topic=c(4),
              method="continuous", moderator="cp", xaxt='n', xlim=c(-60,166), moderator.value=1, add=FALSE, ylim=c(-.3,.5),
ylab="Expected Topic Proportion",
              linecol='red', printlegend=FALSE)
 plot.estimateEffect(polcp, covariate="dayscamped", model=pol15cp,topic=c(4),
              method="continuous", moderator="cp", xaxt='n',moderator.value=2, add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(polcp, covariate="dayscamped", model=pol15cp,topic=c(4),
              method="continuous", moderator="cp", moderator.value=3, add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 legend("top", c('Least Committed to Comm Policiing', 'Somewhat Committed', 'Most Committed to Comm Policing'),
      lwd=2, col=c('red', 'blue', 'darkgreen'), bty="n")

 plot.estimateEffect(polcp, covariate="dayscamped", model=pol15cp,topic=c(15),
```

```
            method="continuous", moderator="cp", moderator.value=1, add=FALSE, ylim=c(-.3,.5), ylab="Expected Topic Proportion",
xlab="Days since local camp established",
            xlim=c(-60,166), xaxt='n', linecol='red', printlegend=FALSE)
 plot.estimateEffect(polcp, covariate="dayscamped", model=pol15cp,topic=c(15),
            method="continuous", moderator="cp", moderator.value=2, add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(polcp, covariate="dayscamped", model=pol15cp,topic=c(15),
            method="continuous", moderator="cp", moderator.value=3, add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 mtext("Expected Topic Proportion", 2, line=4)

 plot.estimateEffect(polcp, covariate="dayscamped", model=pol15cp,topic=c(12),
            method="continuous", moderator="cp", moderator.value=1, add=FALSE, ylim=c(-.3,.4), ylab="Expected Topic Proportion",
xlab="Days since local camp established",
            xlim=c(-60,166), linecol='red', printlegend=FALSE)
 plot.estimateEffect(polcp, covariate="dayscamped", model=pol15cp,topic=c(12),
            method="continuous", moderator="cp", moderator.value=2, add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(polcp, covariate="dayscamped", model=pol15cp,topic=c(12),
            method="continuous", moderator="cp", moderator.value=3, add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 mtext("Days Since Local Camp Established", 1, line=3)

}
pdf.f(dd, file= '/Users/nickbadams/Dropbox/NickDiss/Chapter 6 Figures/intCpInds.pdf',
    width=4, height=7)


ee <- function(){
 layout(matrix(1:3, ncol=1, byrow=TRUE))
 par(oma=c(4,6,2,1), mar=c(1,1,1,1), mgp=c(2,1,0),
   cex.axis=1.5)
 plot.estimateEffect(polcp, covariate="dayscamped", model=pol15cp,topic=c(9),
            method="continuous", moderator="cp", xaxt='n', xlim=c(-60,166), moderator.value=1, add=FALSE, ylim=c(-.3,.5),
ylab="Expected Topic Proportion",
            linecol='red', printlegend=FALSE)
 plot.estimateEffect(polcp, covariate="dayscamped", model=pol15cp,topic=c(9),
            method="continuous", moderator="cp", xaxt='n',moderator.value=2, add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(polcp, covariate="dayscamped", model=pol15cp,topic=c(9),
            method="continuous", moderator="cp", moderator.value=3, add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 legend("top", c('Least Committed to Comm Policiing', 'Somewhat Committed', 'Most Committed to Comm Policing'),
      lwd=2, col=c('red', 'blue', 'darkgreen'), bty="n")

 plot.estimateEffect(polcp, covariate="dayscamped", model=pol15cp,topic=c(7),
            method="continuous", moderator="cp", moderator.value=1, add=FALSE, ylim=c(-.3,.5), ylab="Expected Topic Proportion",
xlab="Days since local camp established",
            xlim=c(-60,166), xaxt='n', linecol='red', printlegend=FALSE)
 plot.estimateEffect(polcp, covariate="dayscamped", model=pol15cp,topic=c(7),
            method="continuous", moderator="cp", moderator.value=2, add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(polcp, covariate="dayscamped", model=pol15cp,topic=c(7),
            method="continuous", moderator="cp", moderator.value=3, add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 mtext("Expected Topic Proportion", 2, line=4)

 plot.estimateEffect(polcp, covariate="dayscamped", model=pol15cp,topic=c(8),
            method="continuous", moderator="cp", moderator.value=1, add=FALSE, ylim=c(-.3,.4), ylab="Expected Topic Proportion",
xlab="Days since local camp established",
            xlim=c(-60,166), linecol='red', printlegend=FALSE)
 plot.estimateEffect(polcp, covariate="dayscamped", model=pol15cp,topic=c(8),
            method="continuous", moderator="cp", moderator.value=2, add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(polcp, covariate="dayscamped", model=pol15cp,topic=c(8),
            method="continuous", moderator="cp", moderator.value=3, add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 mtext("Days Since Local Camp Established", 1, line=3)
```

```
}
pdf.f(ee, file= '/Users/nickbadams/Dropbox/NickDiss/Chapter 6 Figures/intCpCamps.pdf',
    width=4, height=7)

####################################
##########################################################################
##########################################################################
##########################################################################
##########################################################################
##########################################################################
##########################################################################
##########################################################################
####################################


pol15xvio <- stm(out$documents,out$vocab,K=15,prevalence =~ pos1 + stability2 + pool2 + nonwhite + budget + crime*s(dayscamped)
+ officers + cp, max.em.its=250, data=out$meta, seed=1867757)

####################################
####################################

polvio <- estimateEffect(c(4, 15, 12, 9, 7, 8, 10) ~ pos1 + stability2 + cp + nonwhite + budget + crime*s(dayscamped) + officers + pool2,
pol15xvio,
                metadata=meta, uncertainty="None")

ff <- function(){
 layout(matrix(1:3, ncol=1, byrow=TRUE))
 par(oma=c(4,6,2,1), mar=c(1,1,1,1), mgp=c(2,1,0),
   cex.axis=1.5)
 plot.estimateEffect(polvio, covariate="dayscamped", model=pol15xvio,topic=c(4),
            method="continuous", moderator="crime", xaxt='n', xlim=c(-60,166), moderator.value=1, add=FALSE, ylim=c(-.3,.5),
ylab="Expected Topic Proportion",
            linecol='red', printlegend=FALSE)
 plot.estimateEffect(polvio, covariate="dayscamped", model=pol15xvio,topic=c(4),
            method="continuous", moderator="crime", xaxt='n',moderator.value=2, add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(polvio, covariate="dayscamped", model=pol15xvio,topic=c(4),
            method="continuous", moderator="crime", moderator.value=3, add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 legend("top", c('Least Violent Crime', 'Modal Violent Crime', 'Most Violent Crime'),
      lwd=2, col=c('red', 'blue', 'darkgreen'), bty="n")

 plot.estimateEffect(polvio, covariate="dayscamped", model=pol15xvio,topic=c(15),
            method="continuous", moderator="crime", moderator.value=1, add=FALSE, ylim=c(-.3,.5), ylab="Expected Topic Proportion",
xlab="Days since local camp established",
            xlim=c(-60,166), xaxt='n', linecol='red', printlegend=FALSE)
 plot.estimateEffect(polvio, covariate="dayscamped", model=pol15xvio,topic=c(15),
            method="continuous", moderator="crime", moderator.value=2, add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(polvio, covariate="dayscamped", model=pol15xvio,topic=c(15),
            method="continuous", moderator="crime", moderator.value=3, add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 mtext("Expected Topic Proportion", 2, line=4)

 plot.estimateEffect(polvio, covariate="dayscamped", model=pol15xvio,topic=c(12),
            method="continuous", moderator="crime", moderator.value=1, add=FALSE, ylim=c(-.3,.4), ylab="Expected Topic Proportion",
xlab="Days since local camp established",
            xlim=c(-60,166), linecol='red', printlegend=FALSE)
 plot.estimateEffect(polvio, covariate="dayscamped", model=pol15xvio,topic=c(12),
            method="continuous", moderator="crime", moderator.value=2, add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(polvio, covariate="dayscamped", model=pol15xvio,topic=c(12),
            method="continuous", moderator="crime", moderator.value=3, add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 mtext("Days Since Local Camp Established", 1, line=3)

}
pdf.f(ff, file= '/Users/nickbadams/Dropbox/NickDiss/Chapter 6 Figures/intViolentInds.pdf',
    width=4, height=7)
```

```
gg <- function(){
 layout(matrix(1:3, ncol=1, byrow=TRUE))
 par(oma=c(4,6,2,1), mar=c(1,1,1,1), mgp=c(2,1,0),
    cex.axis=1.5)
 plot.estimateEffect(polvio, covariate="dayscamped", model=pol15xvio,topic=c(9),
              method="continuous", moderator="crime", xaxt='n', xlim=c(-60,166), moderator.value=1, add=FALSE, ylim=c(-.3,.5),
ylab="Expected Topic Proportion",
              linecol='red', printlegend=FALSE)
 plot.estimateEffect(polvio, covariate="dayscamped", model=pol15xvio,topic=c(9),
              method="continuous", moderator="crime", xaxt='n',moderator.value=2, add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(polvio, covariate="dayscamped", model=pol15xvio,topic=c(9),
              method="continuous", moderator="crime", moderator.value=3, add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 legend("top", c('Least Violent Crime', 'Modal Violent Crime', 'Most Violent Crime'),
      lwd=2, col=c('red', 'blue', 'darkgreen'), bty="n")

 plot.estimateEffect(polvio, covariate="dayscamped", model=pol15xvio,topic=c(7),
              method="continuous", moderator="crime", moderator.value=1, add=FALSE, ylim=c(-.3,.5), ylab="Expected Topic Proportion",
xlab="Days since local camp established",
              xlim=c(-60,166), xaxt='n', linecol='red', printlegend=FALSE)
 plot.estimateEffect(polvio, covariate="dayscamped", model=pol15xvio,topic=c(7),
              method="continuous", moderator="crime", moderator.value=2, add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(polvio, covariate="dayscamped", model=pol15xvio,topic=c(7),
              method="continuous", moderator="crime", moderator.value=3, add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 mtext("Expected Topic Proportion", 2, line=4)

 plot.estimateEffect(polvio, covariate="dayscamped", model=pol15xvio,topic=c(8),
              method="continuous", moderator="crime", moderator.value=1, add=FALSE, ylim=c(-.3,.4), ylab="Expected Topic Proportion",
xlab="Days since local camp established",
              xlim=c(-60,166), linecol='red', printlegend=FALSE)
 plot.estimateEffect(polvio, covariate="dayscamped", model=pol15xvio,topic=c(8),
              method="continuous", moderator="crime", moderator.value=2, add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(polvio, covariate="dayscamped", model=pol15xvio,topic=c(8),
              method="continuous", moderator="crime", moderator.value=3, add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 mtext("Days Since Local Camp Established", 1, line=3)

}
pdf.f(gg, file= '/Users/nickbadams/Dropbox/NickDiss/Chapter 6 Figures/intViolentCamps.pdf',
    width=4, height=7)



####################################
####################################
####################################
####################################
####################################
pol15xnw <- stm(out$documents,out$vocab,K=15,prevalence =~ pos1 + stability2 + pool2 + nonwhite*s(dayscamped) + budget + crime
+ officers + cp, max.em.its=250, data=out$meta, seed=1867757)

####################################
####################################

polnw <- estimateEffect(c(4, 15, 12, 9, 7, 8, 10) ~ pos1 + stability2 + cp + nonwhite*s(dayscamped) + budget + crime + officers + pool2,
pol15xnw,
                metadata=meta, uncertainty="None")

hh <- function(){
 layout(matrix(1:3, ncol=1, byrow=TRUE))
 par(oma=c(4,6,2,1), mar=c(1,1,1,1), mgp=c(2,1,0),
    cex.axis=1.5)
 plot.estimateEffect(polnw, covariate="dayscamped", model=pol15xnw,topic=c(4),
```

```
        method="continuous", moderator="nonwhite", xaxt='n', xlim=c(-60,166), moderator.value=1, add=FALSE, ylim=c(-.7,.9),
ylab="Expected Topic Proportion",
        linecol='red', printlegend=FALSE)
 plot.estimateEffect(polnw, covariate="dayscamped", model=pol15xnw,topic=c(4),
        method="continuous", moderator="nonwhite", xaxt='n',moderator.value=2, add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(polnw, covariate="dayscamped", model=pol15xnw,topic=c(4),
        method="continuous", moderator="nonwhite", moderator.value=3, add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 legend("top", c('Least Diverse Depts', 'Modal Dept Diversity', 'Most Diverse Depts'),
     lwd=2, col=c('red', 'blue', 'darkgreen'), bty="n")

 plot.estimateEffect(polnw, covariate="dayscamped", model=pol15xnw,topic=c(15),
        method="continuous", moderator="nonwhite", moderator.value=1, add=FALSE, ylim=c(-.7,.9), ylab="Expected Topic
Proportion", xlab="Days since local camp established",
        xlim=c(-60,166), xaxt='n', linecol='red', printlegend=FALSE)
 plot.estimateEffect(polnw, covariate="dayscamped", model=pol15xnw,topic=c(15),
        method="continuous", moderator="nonwhite", moderator.value=2, add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(polnw, covariate="dayscamped", model=pol15xnw,topic=c(15),
        method="continuous", moderator="nonwhite", moderator.value=3, add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 mtext("Expected Topic Proportion", 2, line=4)

 plot.estimateEffect(polnw, covariate="dayscamped", model=pol15xnw,topic=c(12),
        method="continuous", moderator="nonwhite", moderator.value=1, add=FALSE, ylim=c(-.7,.9), ylab="Expected Topic
Proportion", xlab="Days since local camp established",
        xlim=c(-60,166), linecol='red', printlegend=FALSE)
 plot.estimateEffect(polnw, covariate="dayscamped", model=pol15xnw,topic=c(12),
        method="continuous", moderator="nonwhite", moderator.value=2, add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(polnw, covariate="dayscamped", model=pol15xnw,topic=c(12),
        method="continuous", moderator="nonwhite", moderator.value=3, add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 mtext("Days Since Local Camp Established", 1, line=3)

}
pdf.f(hh, file= '/Users/nickbadams/Dropbox/NickDiss/Chapter 6 Figures/intnonwhiteInds.pdf',
    width=4, height=7)


ii <- function(){
 layout(matrix(1:3, ncol=1, byrow=TRUE))
 par(oma=c(4,6,2,1), mar=c(1,1,1,1), mgp=c(2,1,0),
   cex.axis=1.5)
 plot.estimateEffect(polnw, covariate="dayscamped", model=pol15xnw,topic=c(9),
        method="continuous", moderator="nonwhite", xaxt='n', xlim=c(-60,166), moderator.value=1, add=FALSE, ylim=c(-.7,.9),
ylab="Expected Topic Proportion",
        linecol='red', printlegend=FALSE)
 plot.estimateEffect(polnw, covariate="dayscamped", model=pol15xnw,topic=c(9),
        method="continuous", moderator="nonwhite", xaxt='n',moderator.value=2, add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(polnw, covariate="dayscamped", model=pol15xnw,topic=c(9),
        method="continuous", moderator="nonwhite", moderator.value=3, add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 legend("top", c('Least Diverse Depts', 'Modal Dept Diversity', 'Most Diverse Depts'),
     lwd=2, col=c('red', 'blue', 'darkgreen'), bty="n")

 plot.estimateEffect(polnw, covariate="dayscamped", model=pol15xnw,topic=c(7),
        method="continuous", moderator="nonwhite", moderator.value=1, add=FALSE, ylim=c(-.7,.9), ylab="Expected Topic
Proportion", xlab="Days since local camp established",
        xlim=c(-60,166), xaxt='n', linecol='red', printlegend=FALSE)
 plot.estimateEffect(polnw, covariate="dayscamped", model=pol15xnw,topic=c(7),
        method="continuous", moderator="nonwhite", moderator.value=2, add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
 plot.estimateEffect(polnw, covariate="dayscamped", model=pol15xnw,topic=c(7),
        method="continuous", moderator="nonwhite", moderator.value=3, add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
 mtext("Expected Topic Proportion", 2, line=4)
```

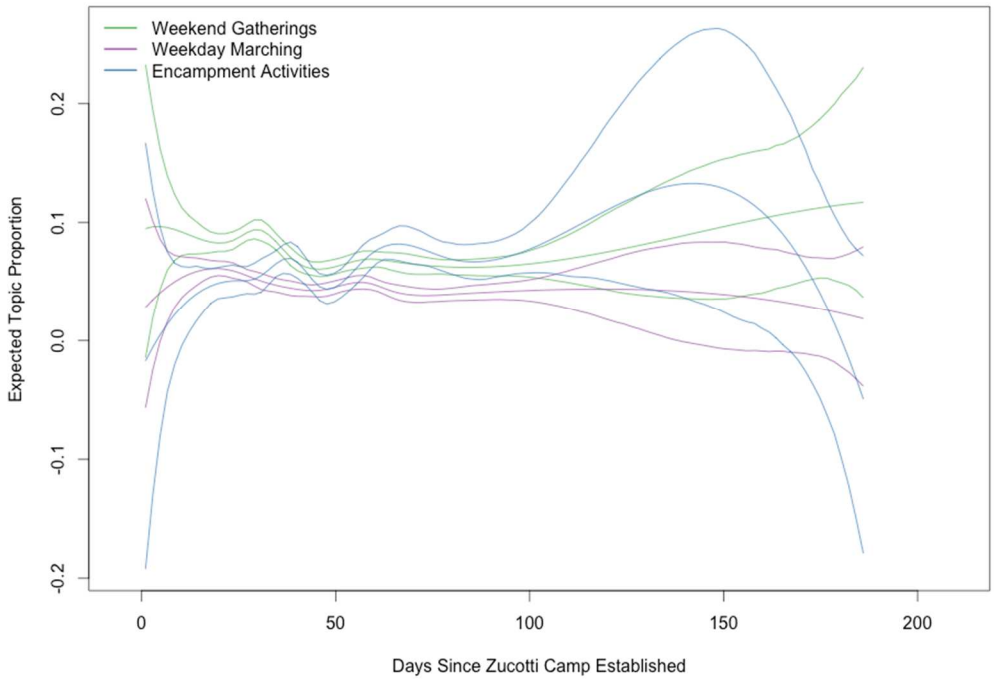
```
  plot.estimateEffect(polnw, covariate="dayscamped", model=pol15xnw,topic=c(8),
                      method="continuous", moderator="nonwhite", moderator.value=1, add=FALSE, ylim=c(-.7,.9), ylab="Expected Topic
Proportion", xlab="Days since local camp established",
                      xlim=c(-60,166), linecol='red', printlegend=FALSE)
  plot.estimateEffect(polnw, covariate="dayscamped", model=pol15xnw,topic=c(8),
                      method="continuous", moderator="nonwhite", moderator.value=2, add=TRUE, labeltype = "custom", linecol='blue',
printlegend=FALSE)
  plot.estimateEffect(polnw, covariate="dayscamped", model=pol15xnw,topic=c(8),
                      method="continuous", moderator="nonwhite", moderator.value=3, add=TRUE, labeltype = "custom", linecol='darkgreen',
printlegend=FALSE)
  mtext("Days Since Local Camp Established", 1, line=3)

}
pdf.f(ii, file= '/Users/nickbadams/Dropbox/NickDiss/Chapter 6 Figures/intnonwhiteCamps.pdf',
     width=4, height=7)
```

Appendix B: Figures Counting Time from Day 0 of New York's Encampment

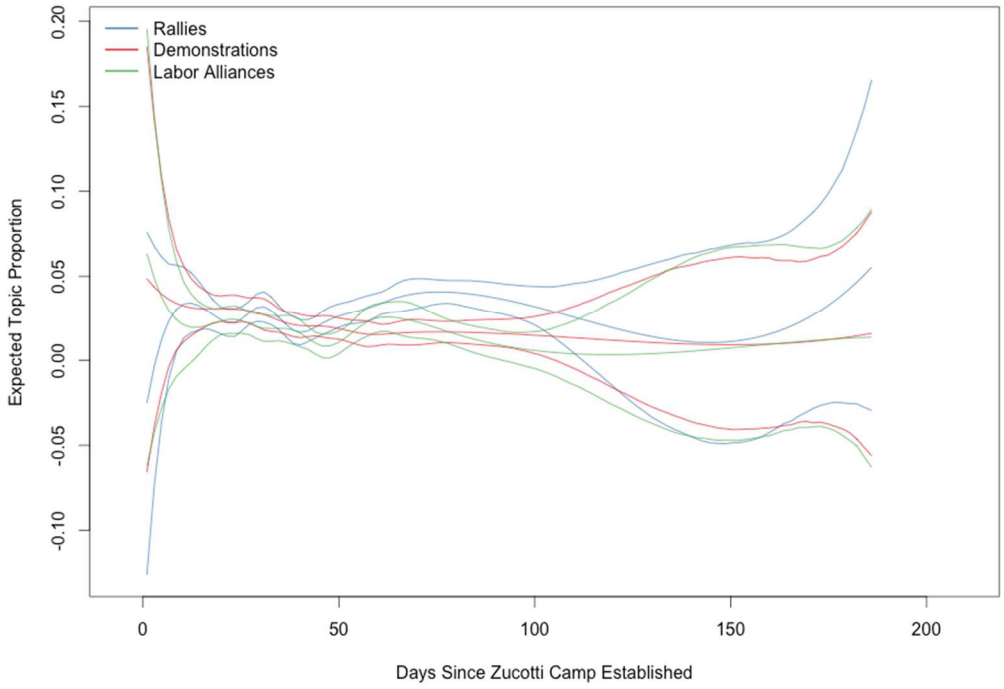
Each of the figures below can be viewed alongside each of the figures of the dissertation. Figure X.4.2, for example, differs only from Figure 4.2 in that *performances* are modeled from Day 0 of the OWS encampment at Zuccotti Park in New York. See Chapter 3 Data Concerns for the motivation behind these figures.

Figure X.4.2 – Weekend Gatherings, Encampment Activities, Weekday Marches



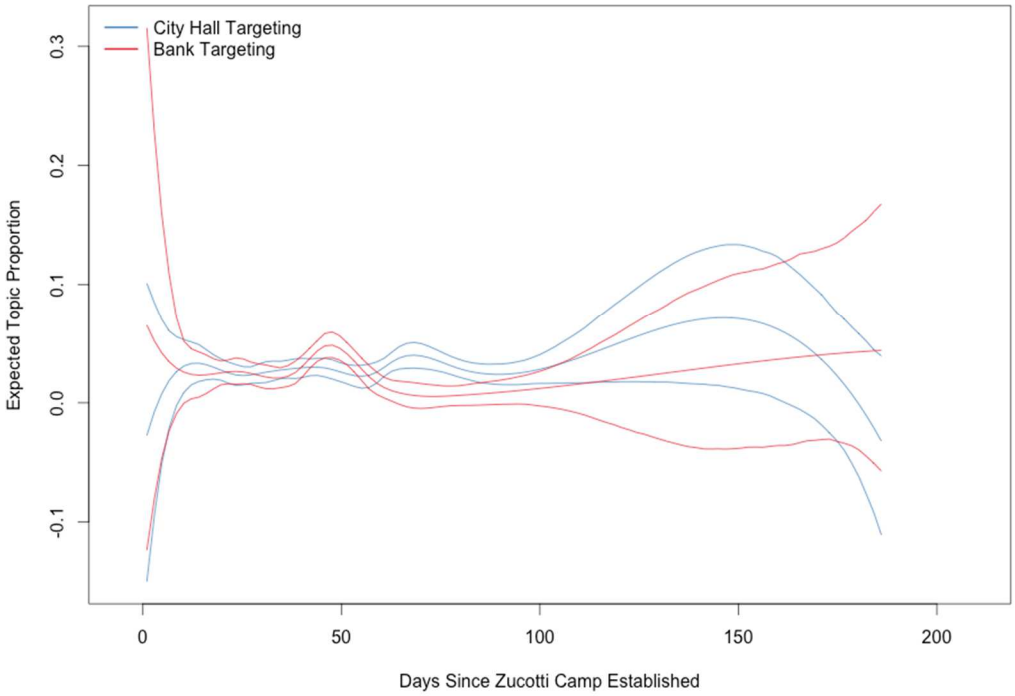
Note: Predicted prevalence of Weekend Gathering, Weekday Marching, and Encampment Activities performances by day since Zuccotti camp established across the full corpus of text units describing protester-initiated contentious gatherings. All POS variables (number of liberals, number of power centers, and political instability) are held at their means.

Figure X.4.3 – Rallies, Demonstrations, and Labor Alliances



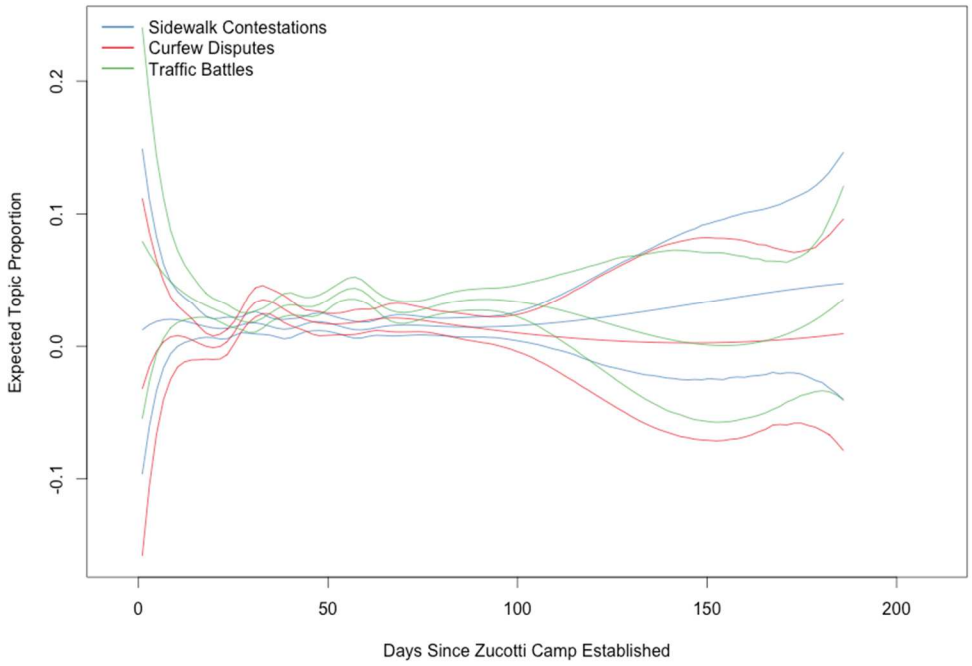
Note: Predicted prevalence of Rallies, Demonstrations, and Labor Alliances performances by day since Zuccotti camp established across the full corpus of text units describing protester-initiated contentious gatherings. All POS variables (number of liberals, number of power centers, and political instability) are held at their means.

Figure X.4.4 – City Hall Targeting and Bank Targeting



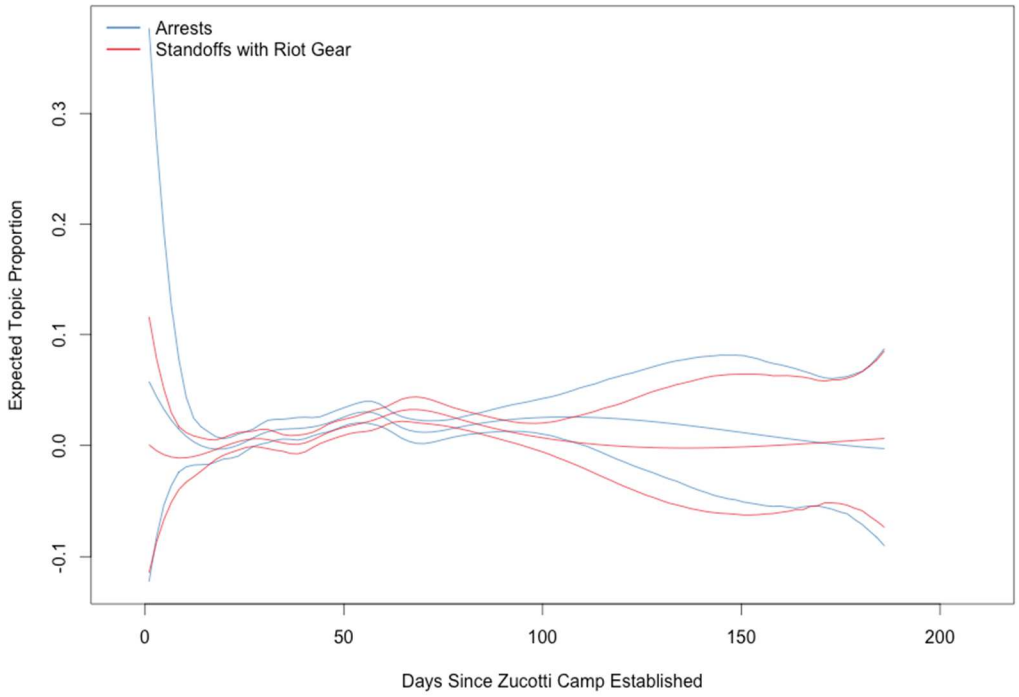
Note: Predicted prevalence of City Hall Targeting and Bank Targeting performances by day since Zuccotti camp established across the full corpus of text units describing protester-initiated contentious gatherings. All POS variables (number of liberals, number of power centers, and political instability) are held at their means.

Figure X.4.5 – Sidewalk Contestation, Curfew Disputes, Traffic Battles



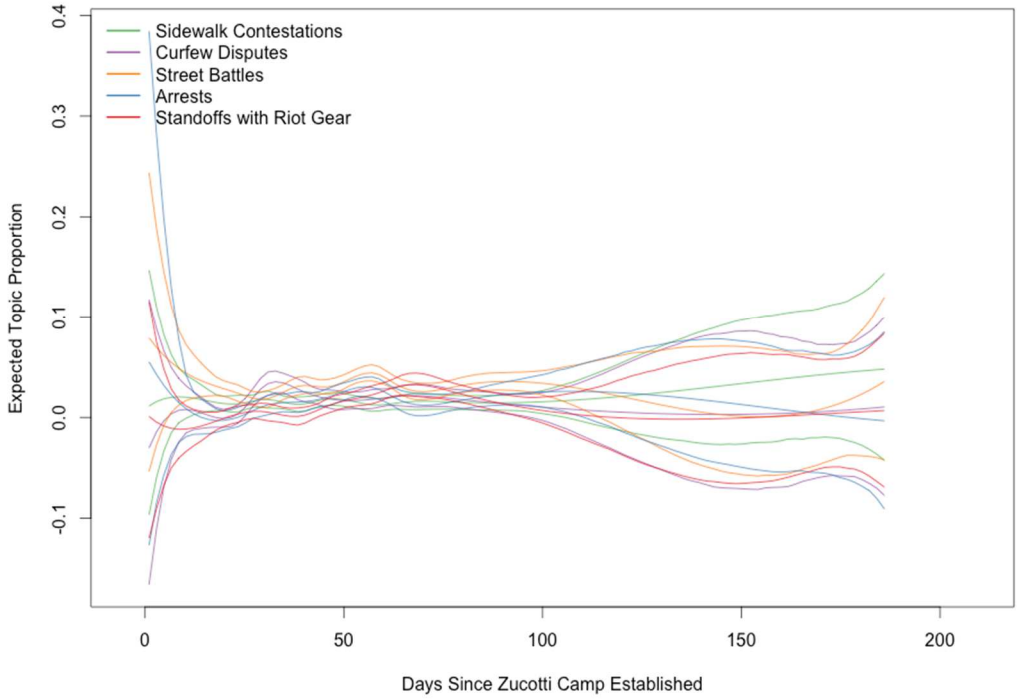
Note: Predicted prevalence of Sidewalk Contestation, Curfew Disputes, and Traffic Battles performances by day since Zuccotti camp established across the full corpus of text units describing protester-initiated contentious gatherings. All POS variables (number of liberals, number of power centers, and political instability) are held at their means.

Figure X.4.6 – Arrests, Standoffs with Riot Gear



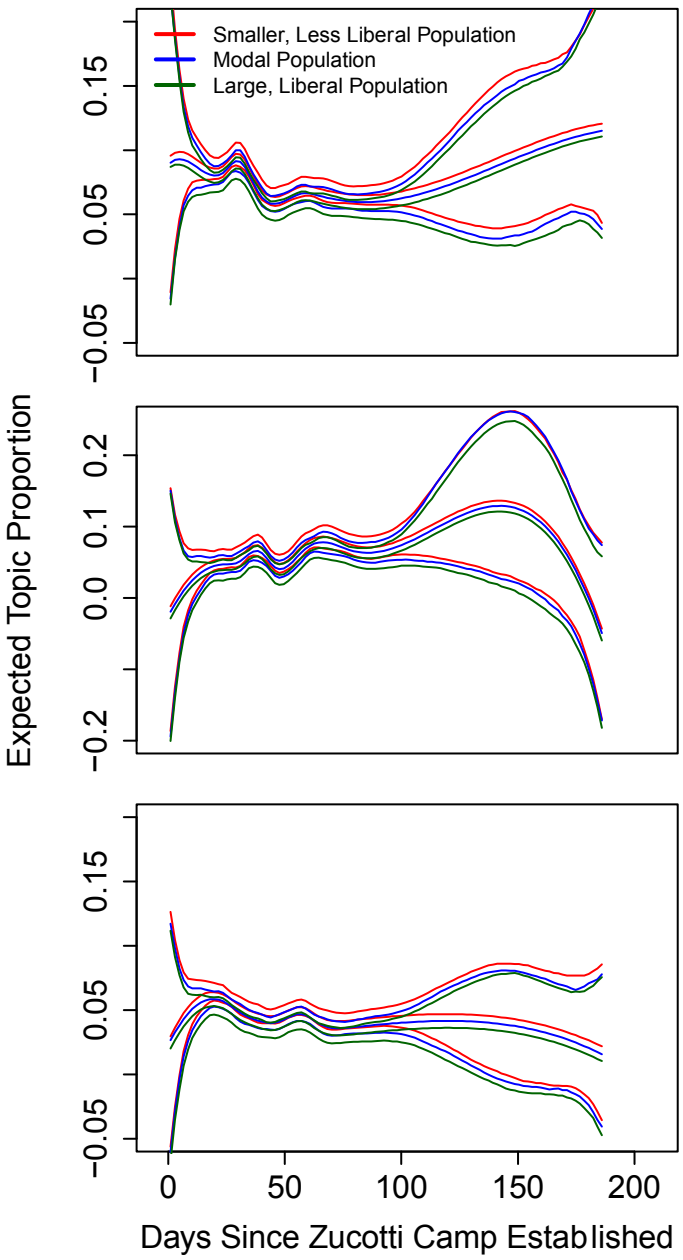
Note: Predicted prevalence of Arrests and Standoffs with Riot Gear performances by day since Zuccotti camp established across the full corpus of text units describing protester-initiated contentious gatherings. All POS variables (number of liberals, number of power centers, and political instability) are held at their means

Figure X.4.7 – Policing Responses to Occupy Performances



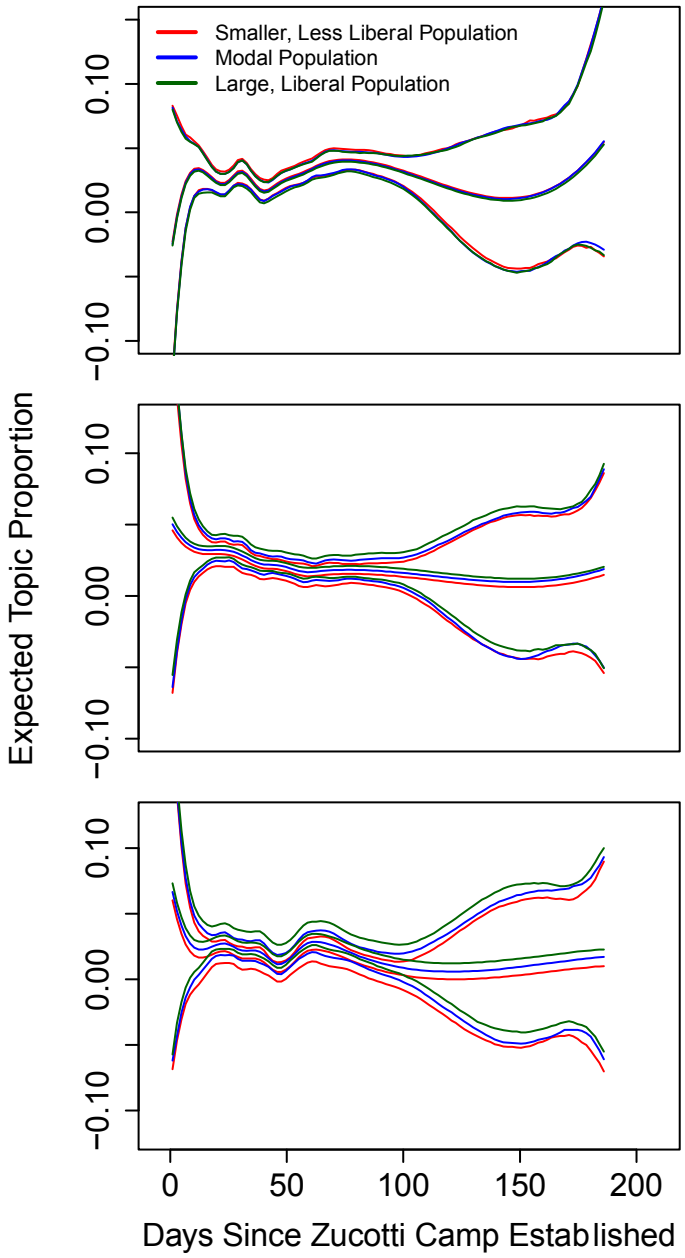
Note: Predicted prevalence of Sidewalk Contestation, Curfew Disputes, Traffic Battles, Arrests and Standoffs with Riot Gear performances by day since Zuccotti camp established across the full corpus of text units describing protester-initiated contentious gatherings. All POS variables (number of liberals, number of power centers, and political instability) are held at their means

Figure X.4.8 – Weekend Gatherings, Encampment Activities, Weekday Marches by Size of Liberal Population



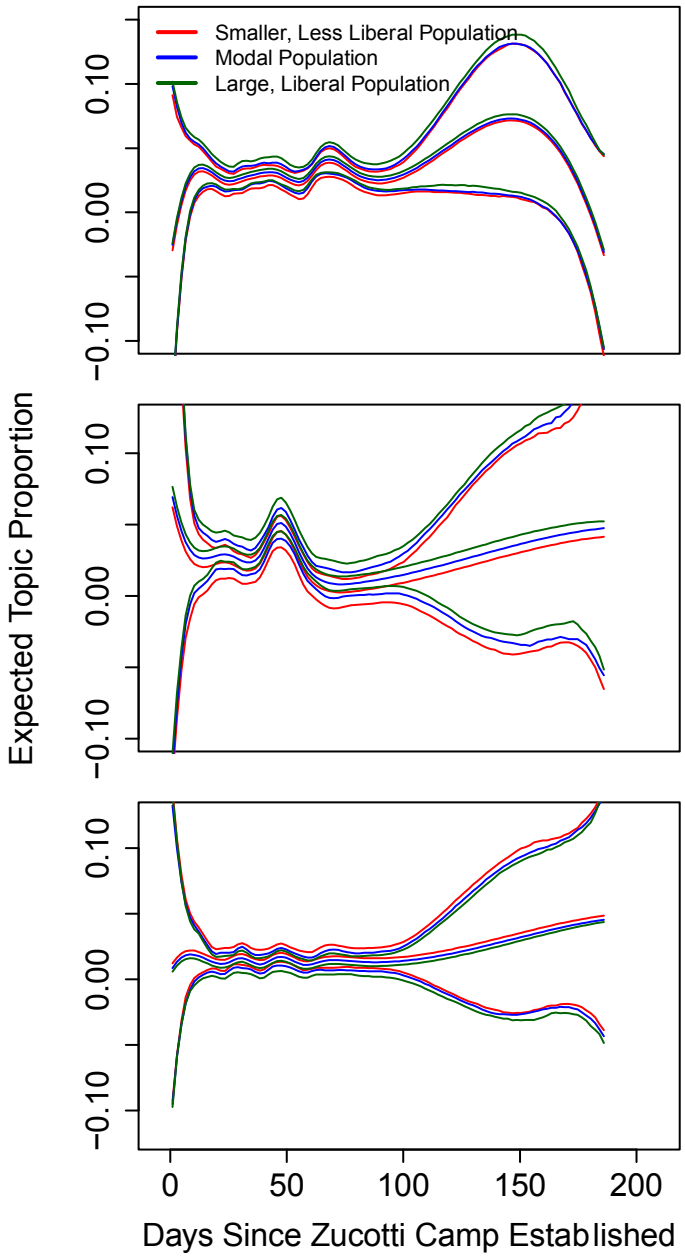
Note: Panel 1 (Top), Panel 2 (Middle), and Panel 3 (Bottom) show the predicted prevalence of Weekend Gatherings, Encampment Activities, and Weekday Marches in the corpus for each day of the Occupy movement. Each panel displays three lines predicting topic prevalence for the highest, lowest, and middle value of the 'number of liberals' variable. See Methodological Appendix for equations used in estimation and prediction.

Figure X.4.9 – Rallies, Demonstrations, and Labor Alliances by Size of Liberal Population



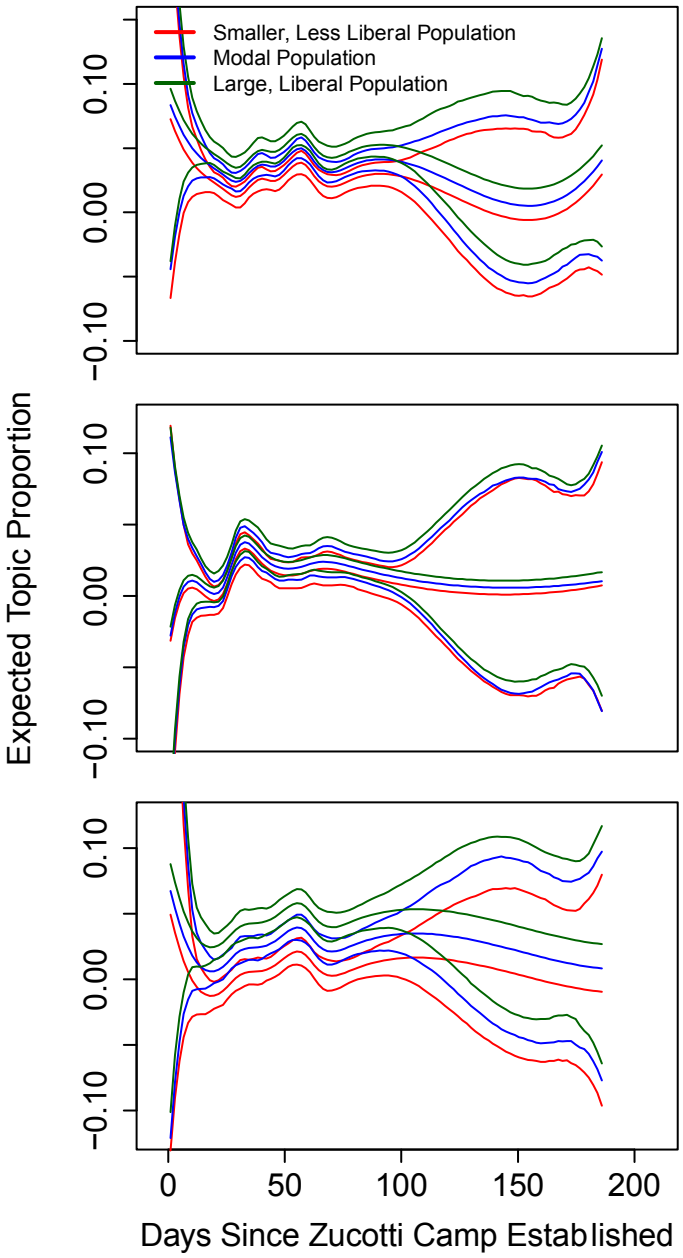
Note: Panel 1 (Top), Panel 2 (Middle), and Panel 3 (Bottom) show the predicted prevalence of Rallies, Demonstrations, and Labor Alliances in the corpus for each day of the Occupy movement. Each panel displays three lines predicting topic prevalence for the highest, lowest, and middle value of the 'number of liberals' variable. See Methodological Appendix for equations used in estimation and prediction.

Figure X.4.10 – City Hall Targeting, Bank Targeting, and Sidewalk Contestation by Size of Liberal Population



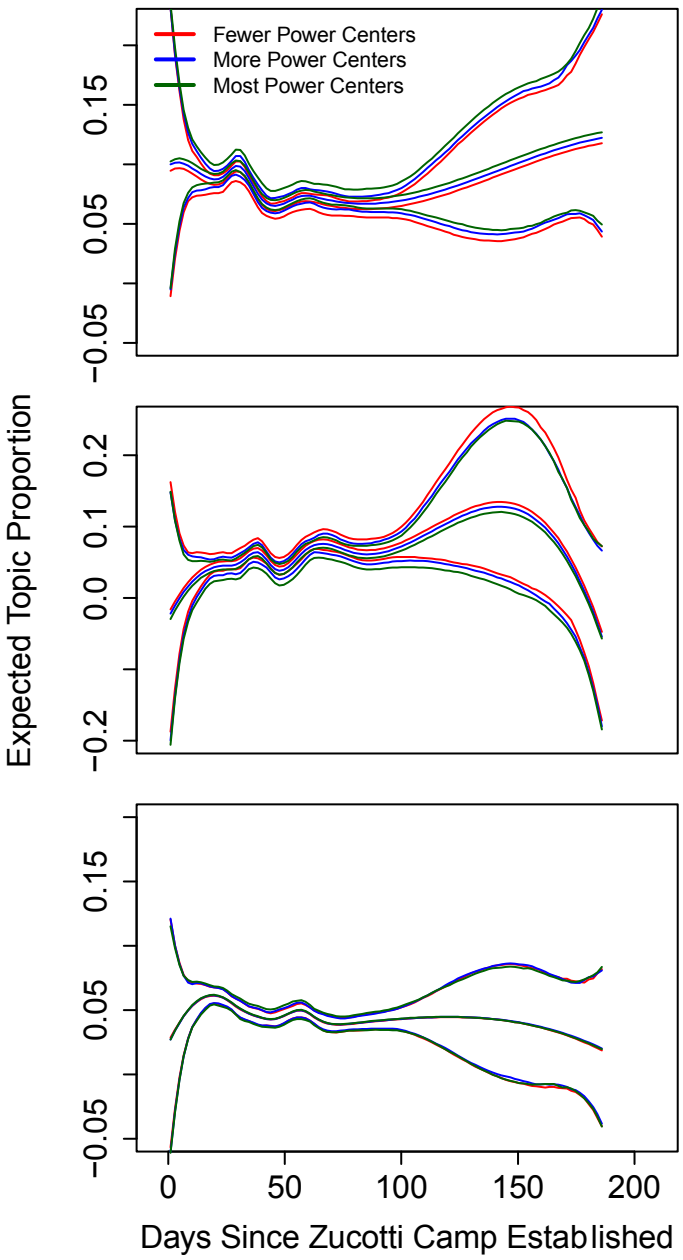
***N**ote: Panel 1 (Top), Panel 2 (Middle), and Panel 3 (Bottom) show the predicted prevalence of City Hall Targeting, Bank Targeting, and Sidewalk Contestation in the corpus for each day of the Occupy movement. Each panel displays three lines predicting topic prevalence for the highest, lowest, and middle value of the 'number of liberals' variable. See Methodological Appendix for equations used in estimation and prediction.*

Figure X.4.11 – Traffic Battles, Curfew Disputes, Arrests by Size of Liberal Population



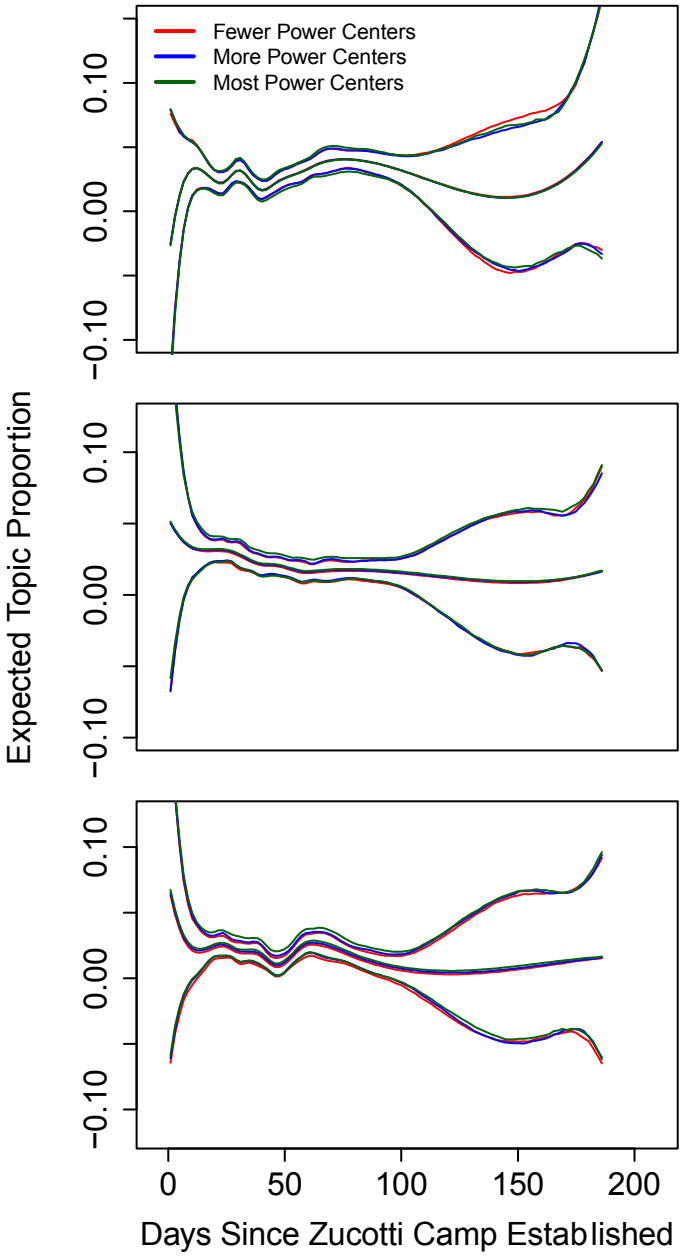
Note: Panel 1 (Top), Panel 2 (Middle), and Panel 3 (Bottom) show the predicted prevalence of Traffic Battles, Curfew Disputes, and Arrests in the corpus for each day of the Occupy movement. Each panel displays three lines predicting topic prevalence for the highest, lowest, and middle value of the 'number of liberals' variable. See Methodological Appendix for equations used in estimation and prediction.

Figure X.5.1 – Weekend Gatherings, Encampment Activities, Weekday Marches by Number of Power Centers



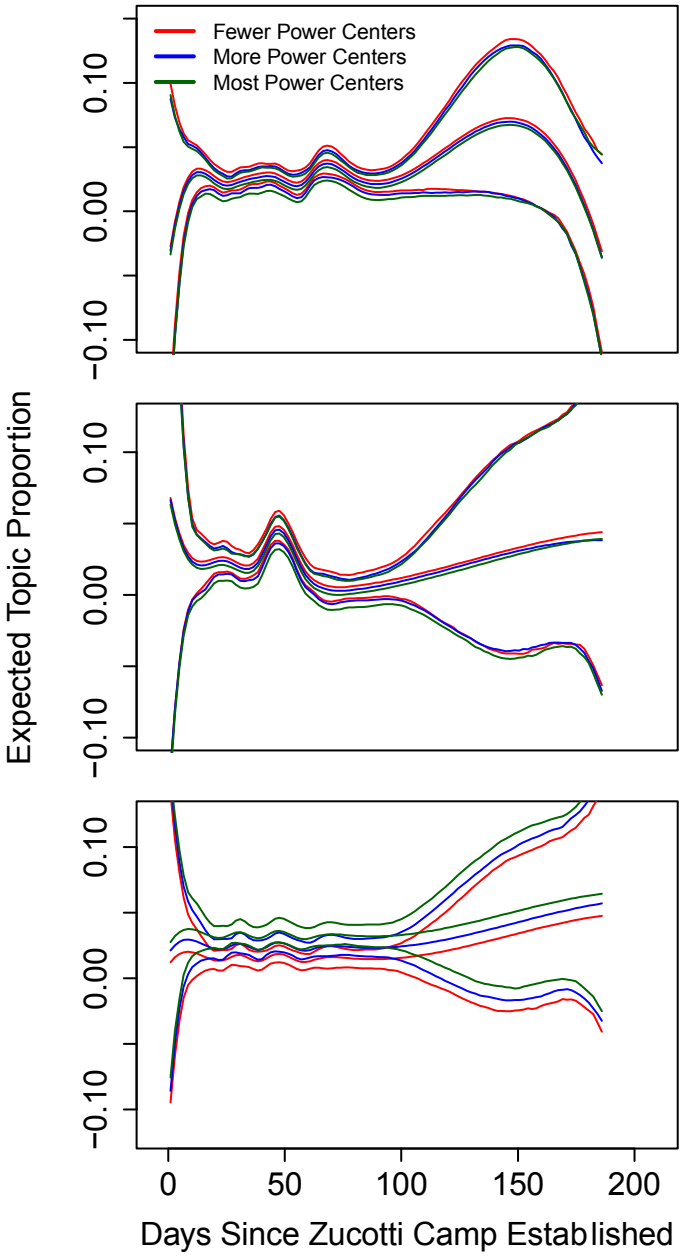
Note: Panel 1 (Top), Panel 2 (Middle), and Panel 3 (Bottom) show the predicted prevalence of Weekend Gatherings, Encampment Activities, Weekday Marches in the corpus for each day of the Occupy movement. Each panel displays three lines predicting topic prevalence for the highest, lowest, and middle value of the 'number of power centers' variable. See Methodological Appendix for equations used in estimation and prediction.

Figure X.5.2 – Rallies, Demonstrations, and Labor Alliances by Number of Power Centers



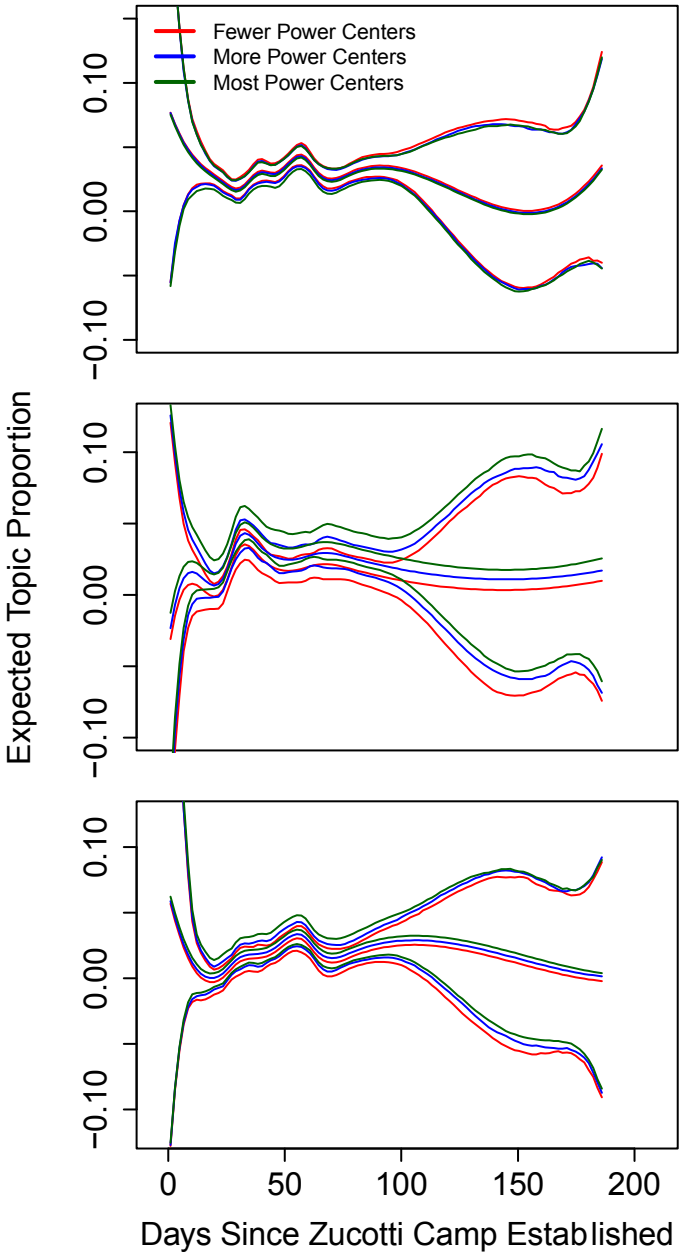
Note: Panel 1 (Top), Panel 2 (Middle), and Panel 3 (Bottom) show the predicted prevalence of Rallies, Demonstrations, and Labor Alliances in the corpus for each day of the Occupy movement. Each panel displays three lines predicting topic prevalence for the highest, lowest, and middle value of the 'number of power centers' variable. See Methodological Appendix for equations used in estimation and prediction.

Figure X.5.3 – City Hall Targeting, Bank Targeting, and Sidewalk Contestation by Number of Power Centers



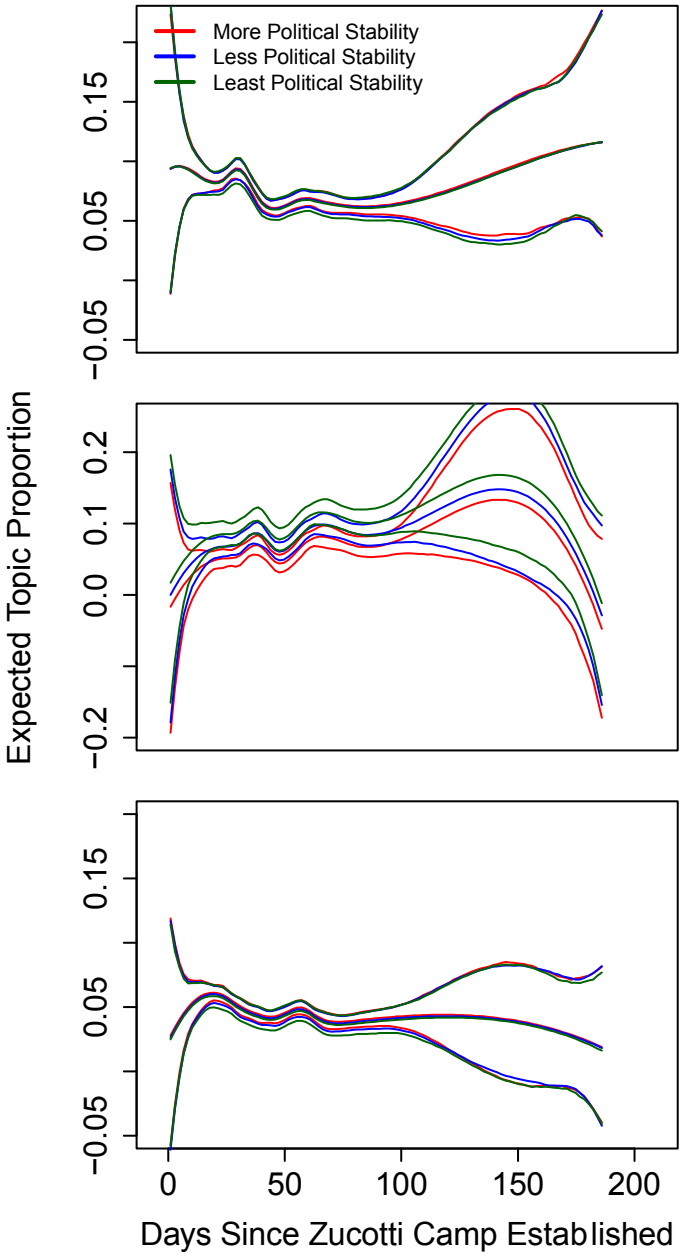
Note: Panel 1 (Top), Panel 2 (Middle), and Panel 3 (Bottom) show the predicted prevalence of City Hall Targeting, Bank Targeting, and Sidewalk Contestation in the corpus for each day of the Occupy movement. Each panel displays three lines predicting topic prevalence for the highest, lowest, and middle value of the 'number of power centers' variable. See Methodological Appendix for equations used in estimation and prediction.

Figure X.5.4 – Traffic Battles, Curfew Disputes, and Arrests by Number of Power Centers



Note: Panel 1 (Top), Panel 2 (Middle), and Panel 3 (Bottom) show the predicted prevalence of Traffic Battles, Curfew Disputes, and Arrests in the corpus for each day of the Occupy movement. Each panel displays three lines predicting topic prevalence for the highest, lowest, and middle value of the 'number of power centers' variable. See Methodological Appendix for equations used in estimation and prediction.

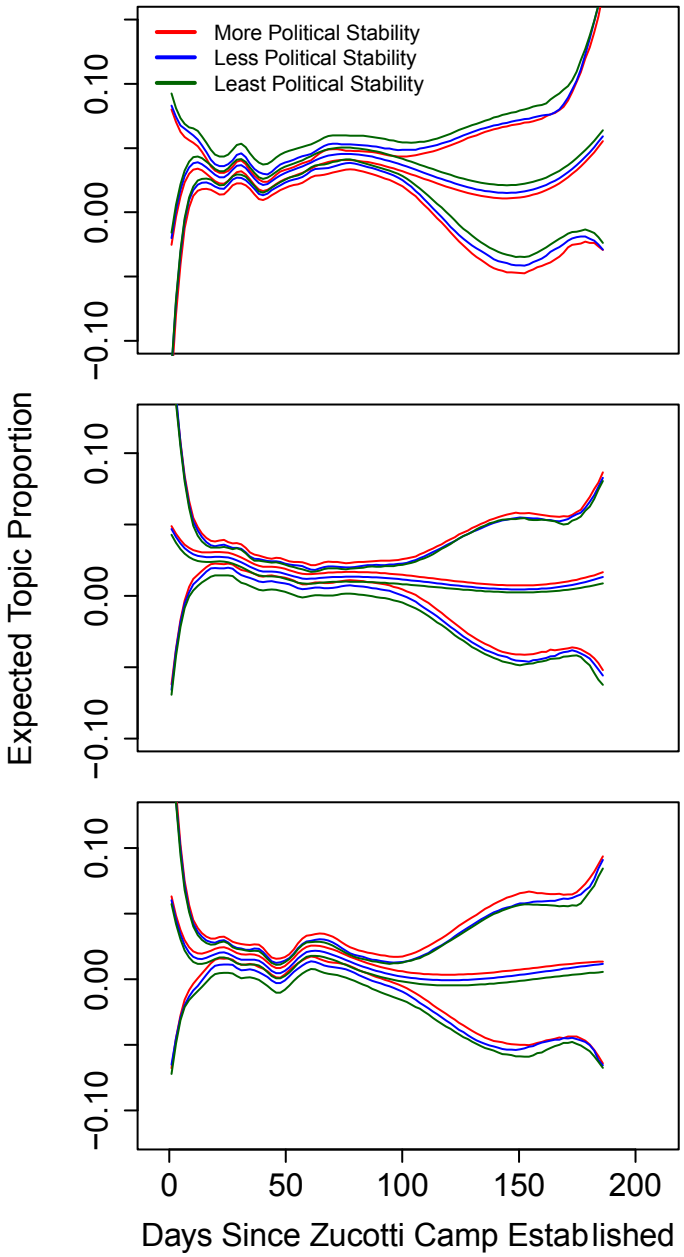
Figure X.5.5 – Weekend Gatherings, Encampment Activities, Weekday Marches by Political Instability



Note: Panel 1 (Top), Panel 2 (Middle), and Panel 3 (Bottom) show the predicted prevalence of Weekend Gatherings, Encampment Activities, and Weekday Marches in the corpus for each day of the Occupy movement. Each panel displays three lines predicting topic prevalence for the highest, lowest, and

middle value of the 'political instability' variable. See Methodological Appendix for equations used in estimation and prediction.

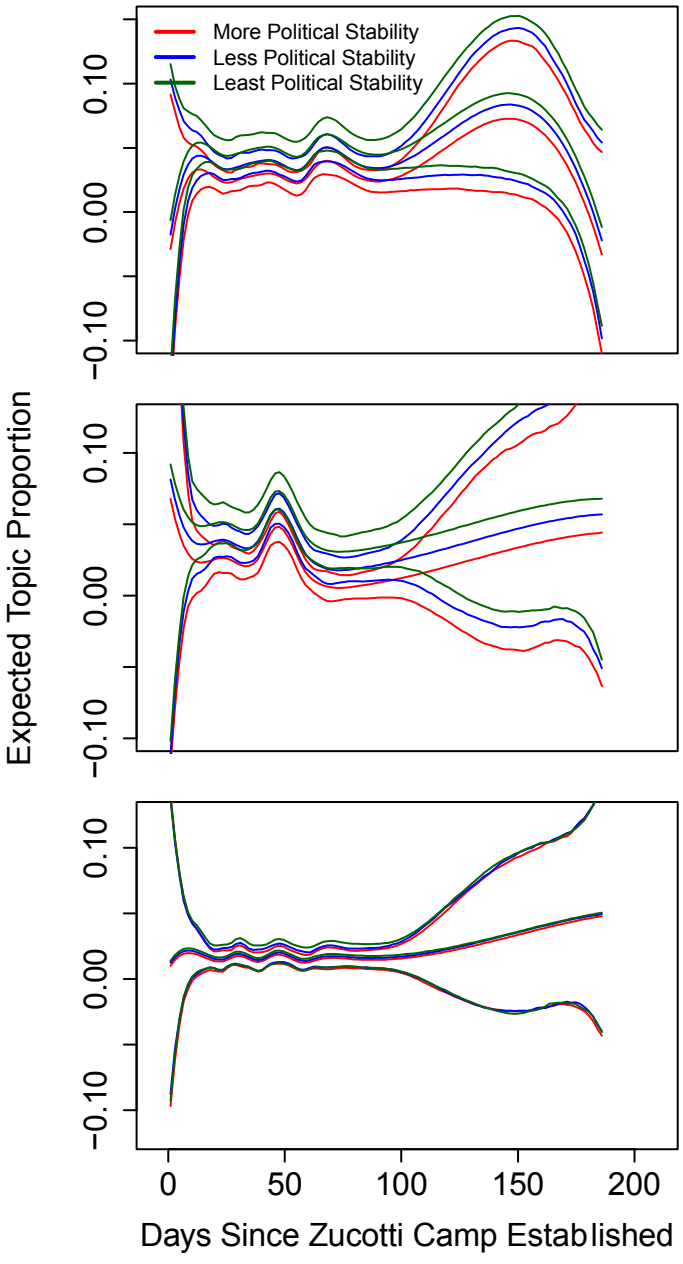
Figure X.5.6 – Rallies, Demonstrations, and Labor Alliances by Political Instability



Note: Panel 1 (Top), Panel 2 (Middle), and Panel 3 (Bottom) show the predicted prevalence of Rallies, Demonstrations, and Labor Alliances in the corpus for each day of the Occupy movement. Each panel displays three lines predicting

topic prevalence for the highest, lowest, and middle value of the 'political instability' variable. See Methodological Appendix for equations used in estimation and prediction.

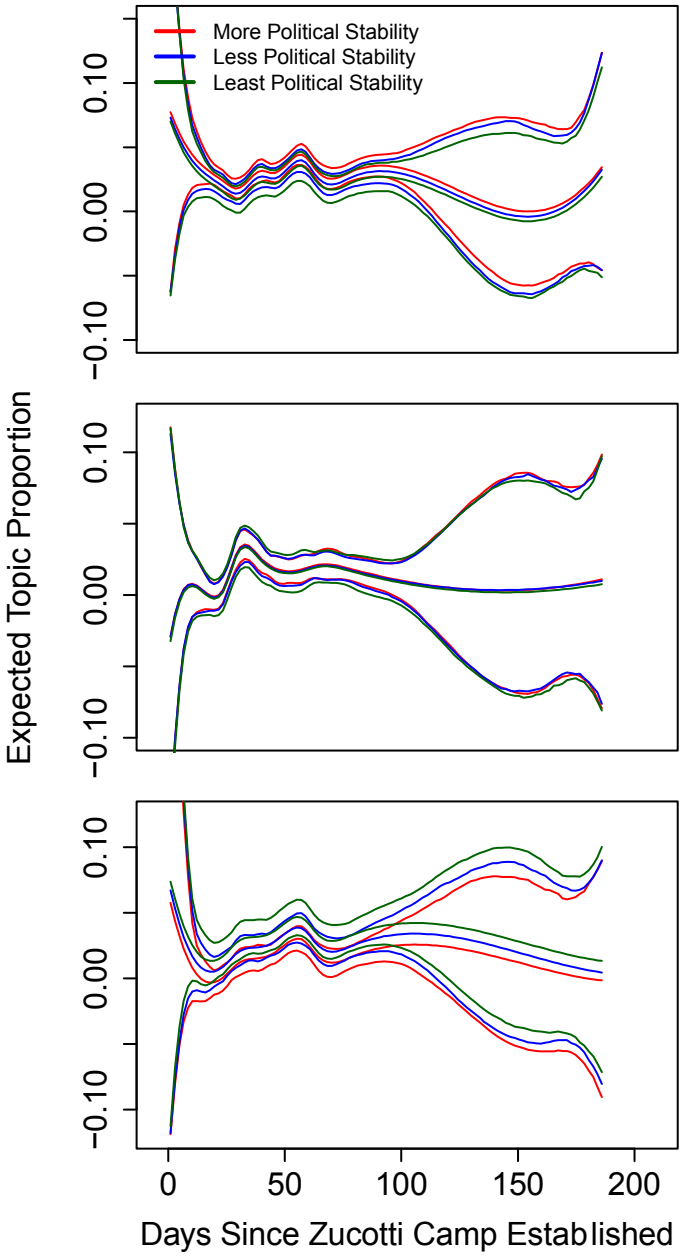
Figure X.5.7 – City Hall Targeting, Bank Targeting, and Sidewalk Contestation



Note: Panel 1 (Top), Panel 2 (Middle), and Panel 3 (Bottom) show the predicted prevalence of City Hall Targeting,

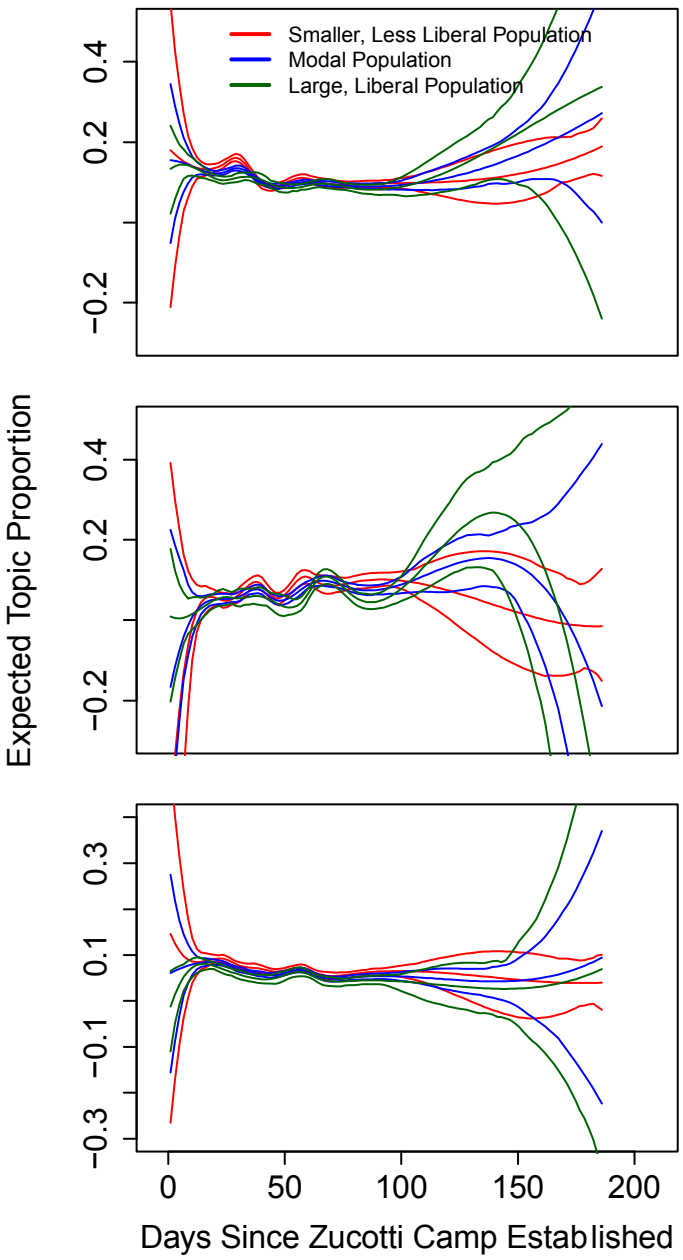
Bank Targeting, and Sidewalk Contestation in the corpus for each day of the Occupy movement. Each panel displays three lines predicting topic prevalence for the highest, lowest, and middle value of the 'political instability' variable. See Methodological Appendix for equations used in estimation and prediction

Figure X.5.8 – Traffic Battles, Curfew Disputes, and Arrests by Political Instability



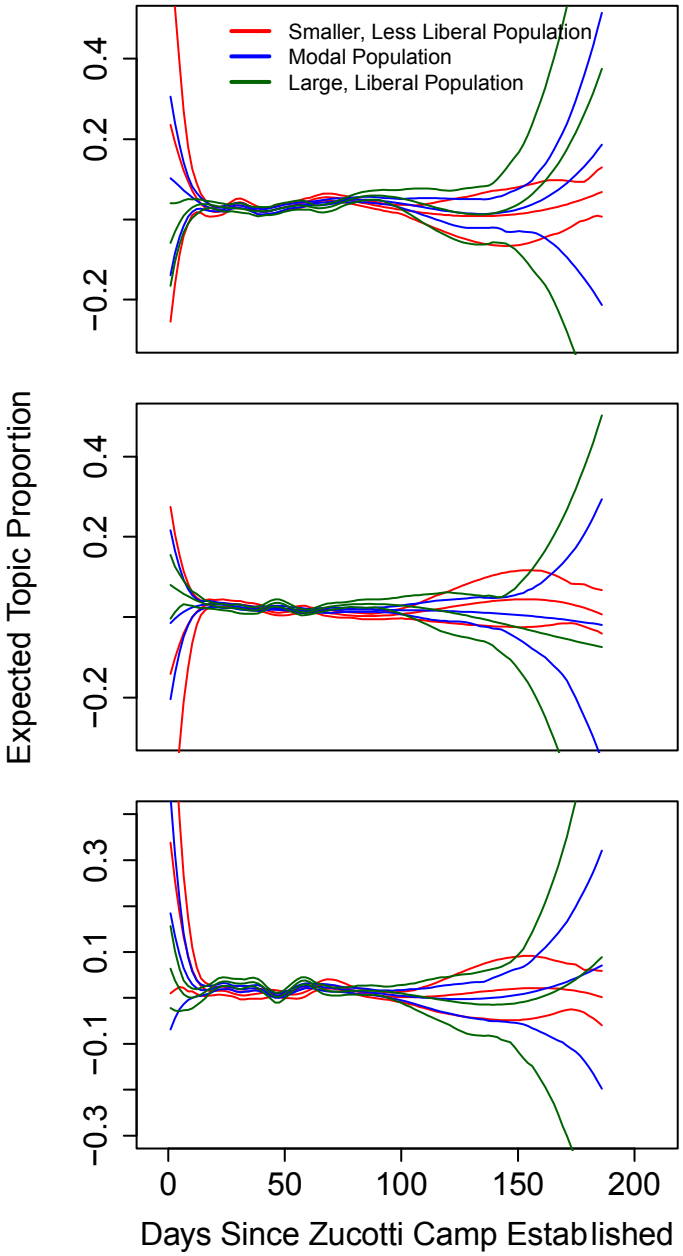
ote: Panel 1 (Top), Panel 2 (Middle), and Panel 3 (Bottom) show the predicted prevalence of Traffic Battles, Curfew Disputes, and Arrests in the corpus for each day of the Occupy movement. Each panel displays three lines predicting topic prevalence for the highest, lowest, and middle value of the 'political instability' variable. See Methodological Appendix for equations used in estimation and prediction.

Figure X.5.9 – Weekend Gatherings, Encampment Activities, Weekday Marches by Size of Liberal Population Interacted with Time



Note: Panel 1 (Top), Panel 2 (Middle), and Panel 3 (Bottom) show the predicted prevalence of Weekend Gatherings, Encampment Activities, Weekday Marches in the corpus for each day of the Occupy movement. Each panel displays three lines predicting topic prevalence for the highest, lowest, and middle value of the 'number of liberals' variable interacted with time. See Methodological Appendix for equations used in estimation and prediction.

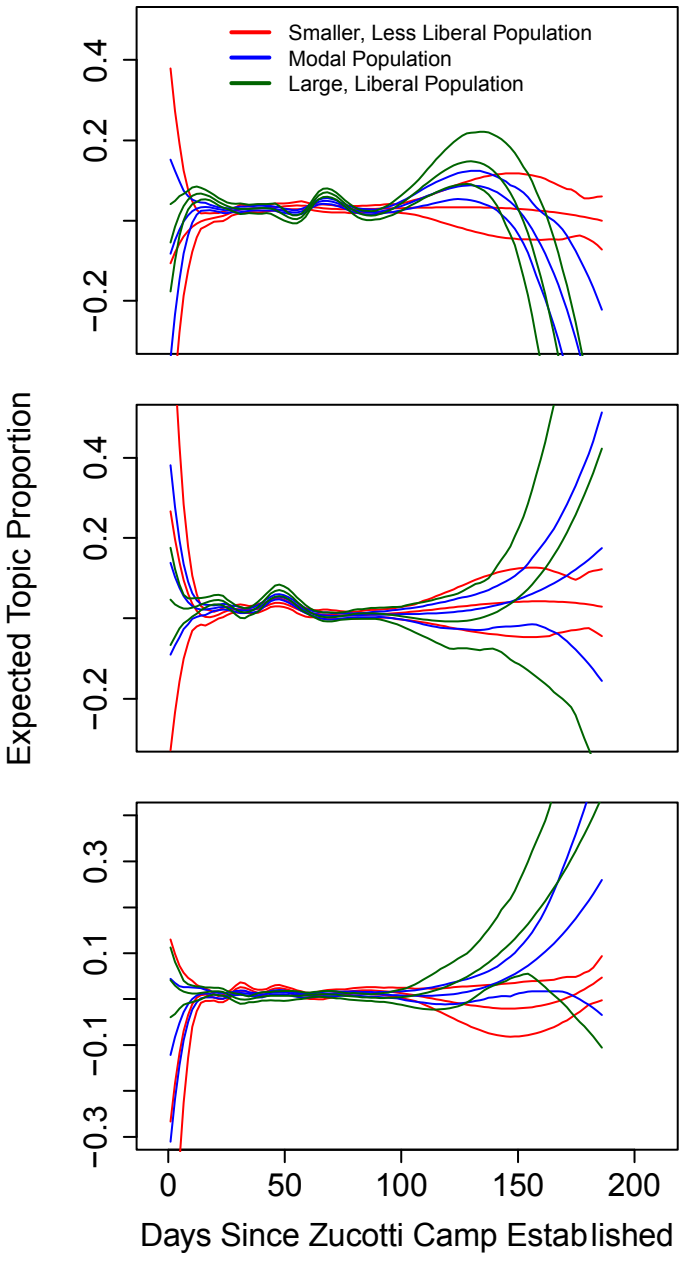
Figure X.5.10 – Rallies, Demonstrations, and Labor Alliances by Size of Liberal Population Interacted with Time



Note: Panel 1 (Top), Panel 2 (Middle), and Panel 3 (Bottom) show the predicted prevalence of Rallies, Demonstrations, and Labor Alliances in the corpus for each day of the Occupy movement. Each panel displays three lines predicting topic prevalence for the highest, lowest, and middle value of the 'number of liberals' variable interacted

with time. See Methodological Appendix for equations used in estimation and prediction.

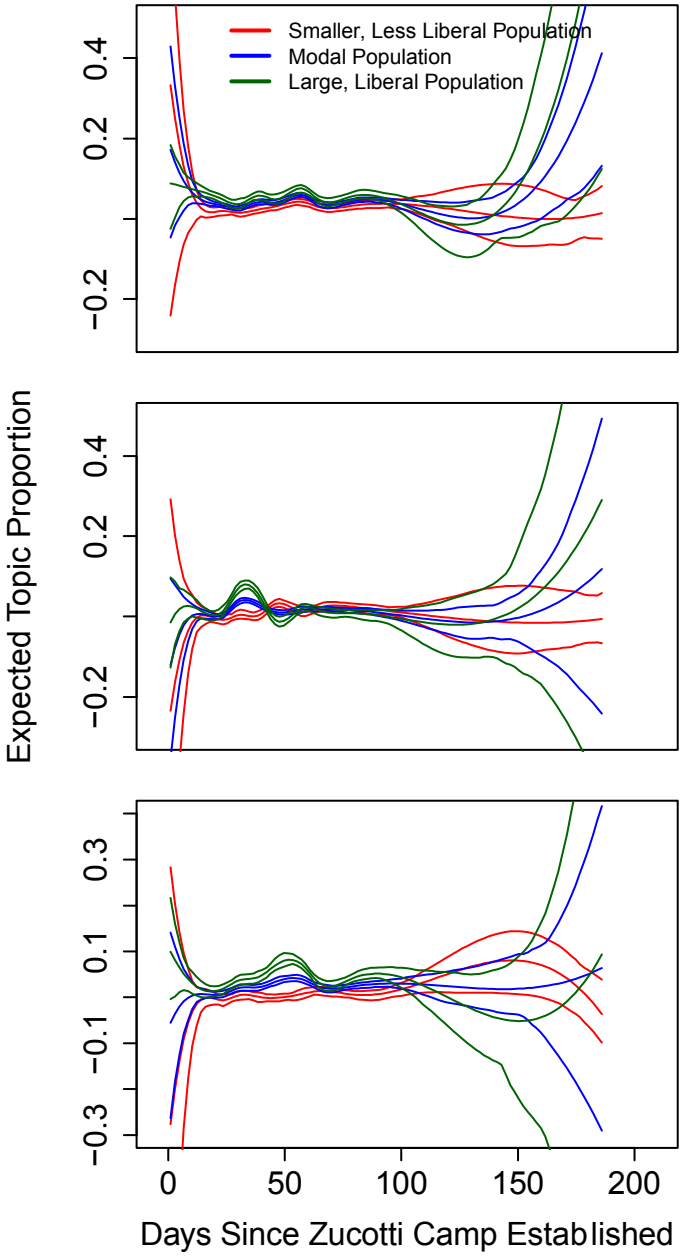
Figure X.5.11 – City Hall Targeting, Bank Targeting, and Sidewalk Contestation by Size of Liberal Population Interacted with Time



Note: Panel 1 (Top), Panel 2 (Middle), and Panel 3 (Bottom) show the predicted prevalence of City Hall Targeting, Bank Targeting, and Sidewalk Contestation in the corpus for each day of the Occupy

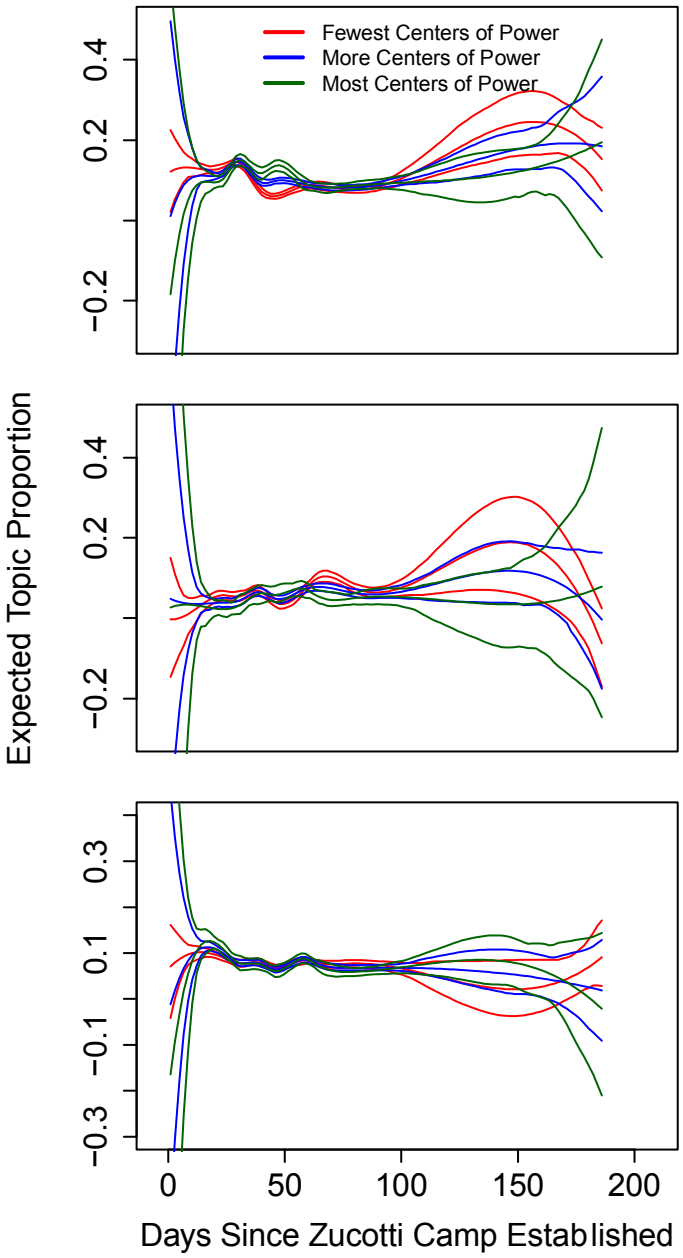
movement. Each panel displays three lines predicting topic prevalence for the highest, lowest, and middle value of the 'number of liberals' variable interacted with time. See Methodological Appendix for equations used in estimation and prediction.

Figure X.5.12 – Traffic Battles, Curfew Disputes, and Arrests by Size of Liberal Population Interacted with Time



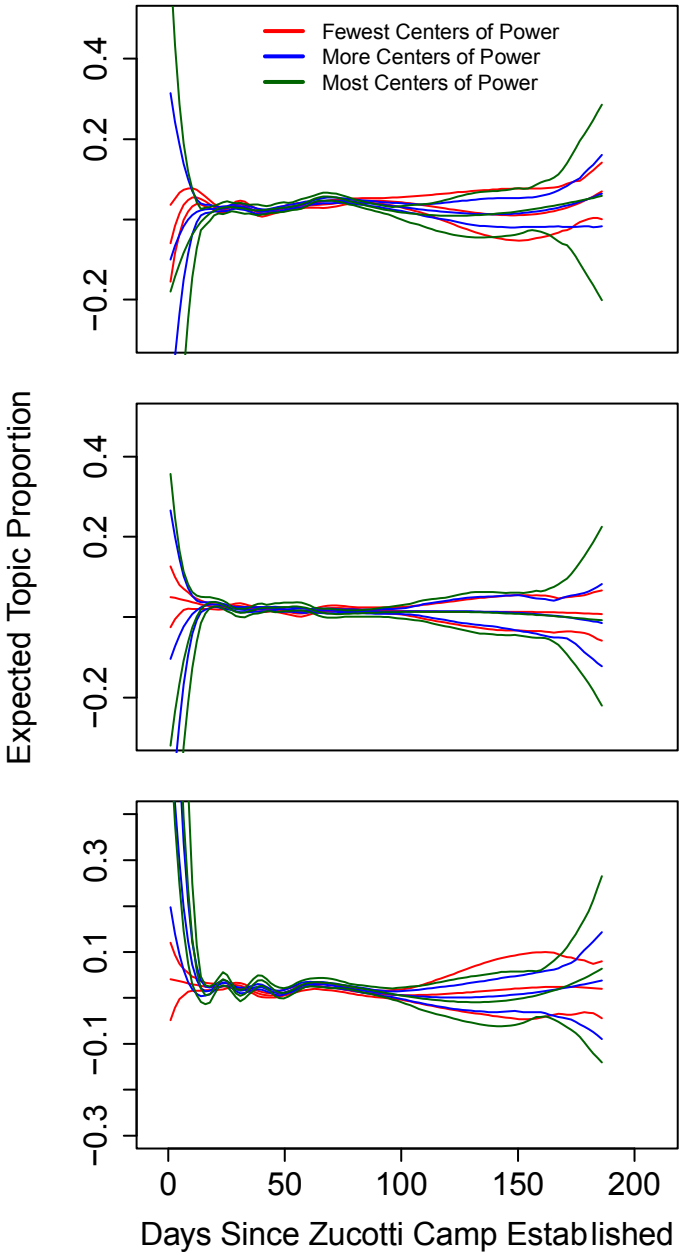
Note: Panel 1 (Top), Panel 2 (Middle), and Panel 3 (Bottom) show the predicted prevalence of Traffic Battles, Curfew Disputes, and Arrests in the corpus for each day of the Occupy movement. Each panel displays three lines predicting topic prevalence for the highest, lowest, and middle value of the 'number of liberals' variable interacted with time. See Methodological Appendix for equations used in estimation and prediction.

Figure X.5.13 – Weekend Gatherings, Encampment Activities, Weekday Marches by Number of Power Centers Interacted with Time



Note: Panel 1 (Top), Panel 2 (Middle), and Panel 3 (Bottom) show the predicted prevalence of Weekend Gatherings, Encampment Activities, Weekday Marches in the corpus for each day of the Occupy movement. Each panel displays three lines predicting topic prevalence for the highest, lowest, and middle value of the 'number of power centers' variable interacted with time. See Methodological Appendix for equations used in estimation and prediction.

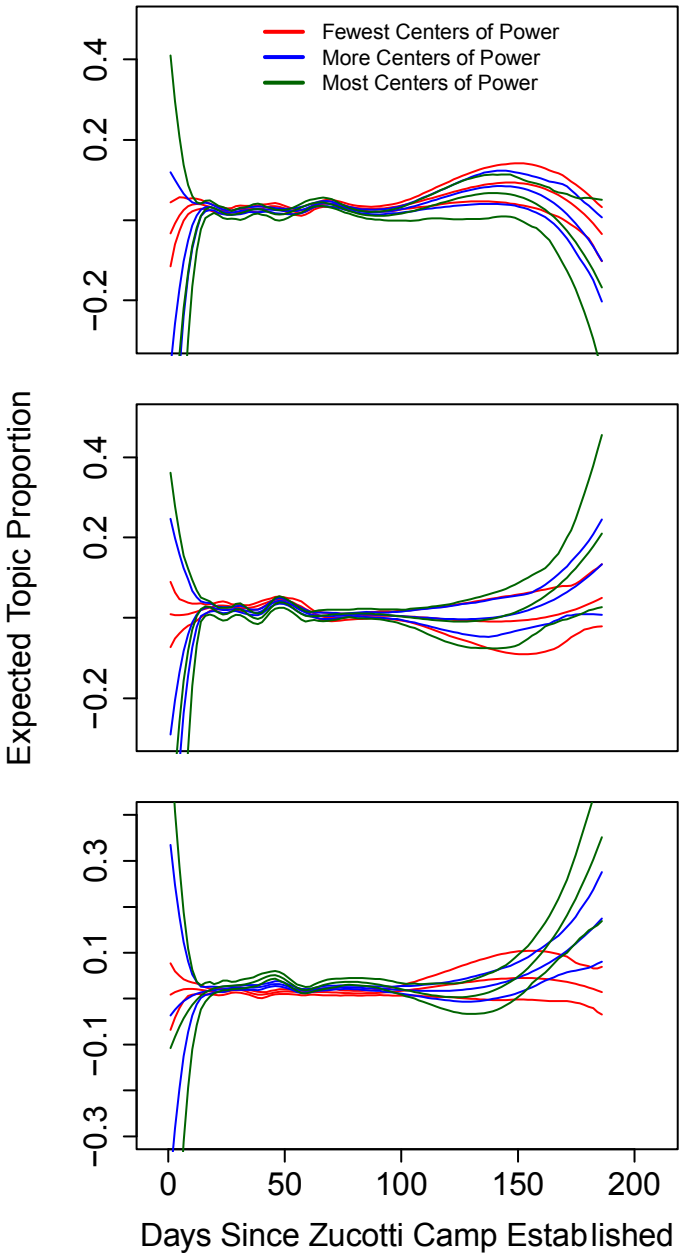
Figure X.5.14 – Rallies, Demonstrations, and Labor Alliances by Number of Power Centers Interacted with Time



Note: Panel 1 (Top), Panel 2 (Middle), and Panel 3 (Bottom) show the predicted prevalence of Rallies, Demonstrations, and Labor Alliances in the corpus for each day of the Occupy movement. Each panel displays three lines predicting topic prevalence for the highest, lowest, and middle value of the 'number of power centers' variable

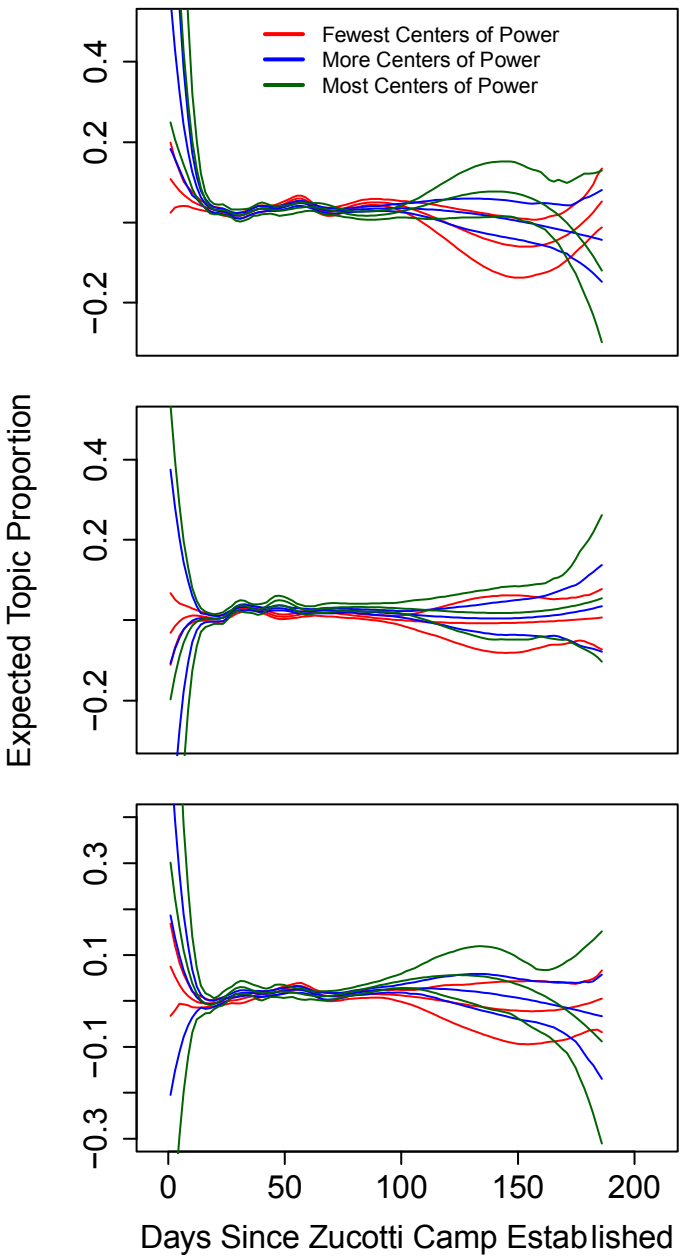
interacted with time. See Methodological Appendix for equations used in estimation and prediction.

Figure X.5.15 – City Hall Targeting, Bank Targeting, and Sidewalk Contestation by Number of Power Centers Interacted with Time



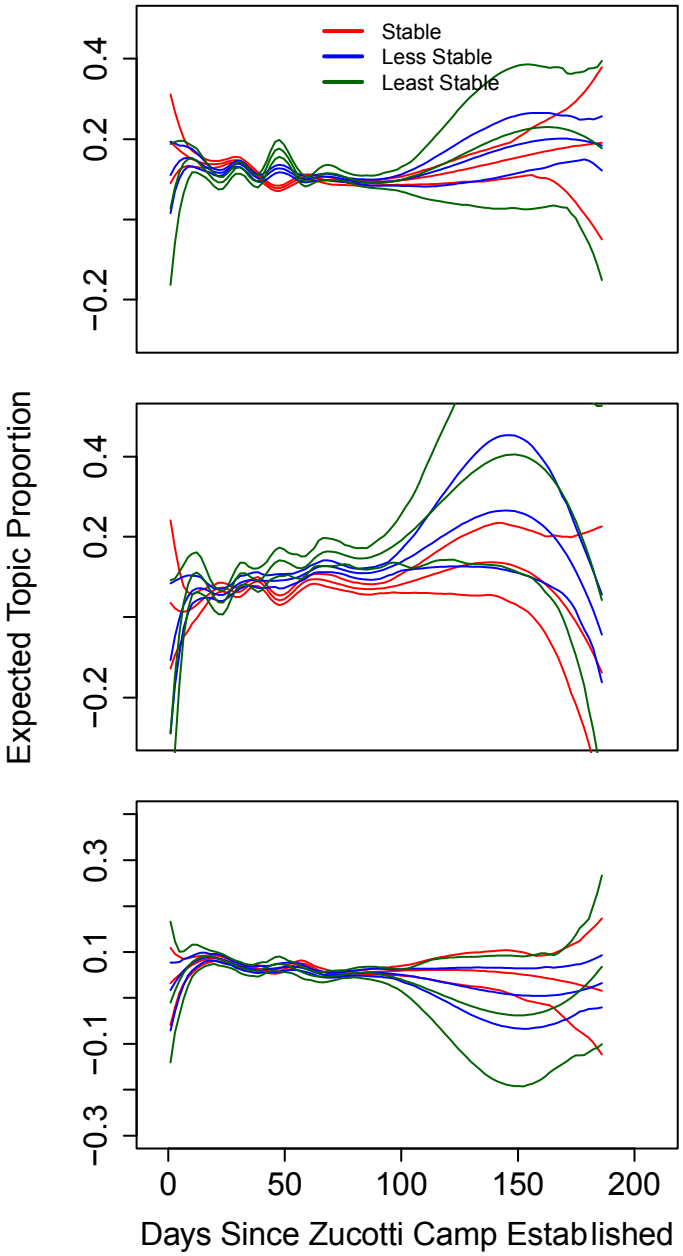
Note: Panel 1 (Top), Panel 2 (Middle), and Panel 3 (Bottom) show the predicted prevalence of City Hall Targeting, Bank Targeting, and Sidewalk Contestation in the corpus for each day of the Occupy movement. Each panel displays three lines predicting topic prevalence for the highest, lowest, and middle value of the 'number of power centers' variable interacted with time. See Methodological Appendix for equations used in estimation and prediction.

Figure X.5.16 – Traffic Battles, Curfew Disputes, and Arrests by Number of Power Centers Interacted with Time



Note: Panel 1 (Top), Panel 2 (Middle), and Panel 3 (Bottom) show the predicted prevalence of Traffic Battles, Curfew Disputes, and Arrests in the corpus for each day of the Occupy movement. Each panel displays three lines predicting topic prevalence for the highest, lowest, and middle value of the 'number of power centers' variable interacted with time. See Methodological Appendix for equations used in estimation and prediction.

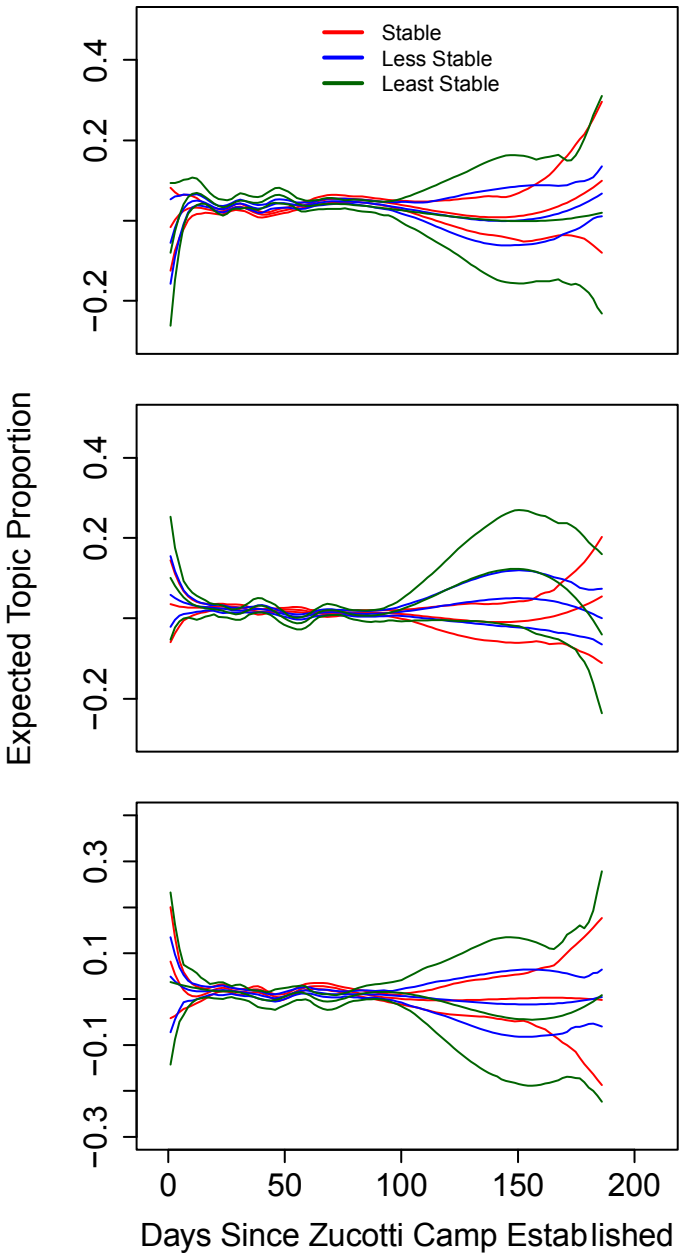
Figure X.5.17 – Weekend Gatherings, Encampment Activities, Weekday Marches by Political Instability Interacted with Time



Note: Panel 1 (Top), Panel 2 (Middle), and Panel 3 (Bottom) show the predicted prevalence of Weekend Gatherings, Encampment Activities, Weekday Marches in the corpus for each day of the Occupy movement. Each panel displays three lines predicting topic prevalence for the highest, lowest, and middle value of the 'political

instability' variable interacted with time. See Methodological Appendix for equations used in estimation and prediction.

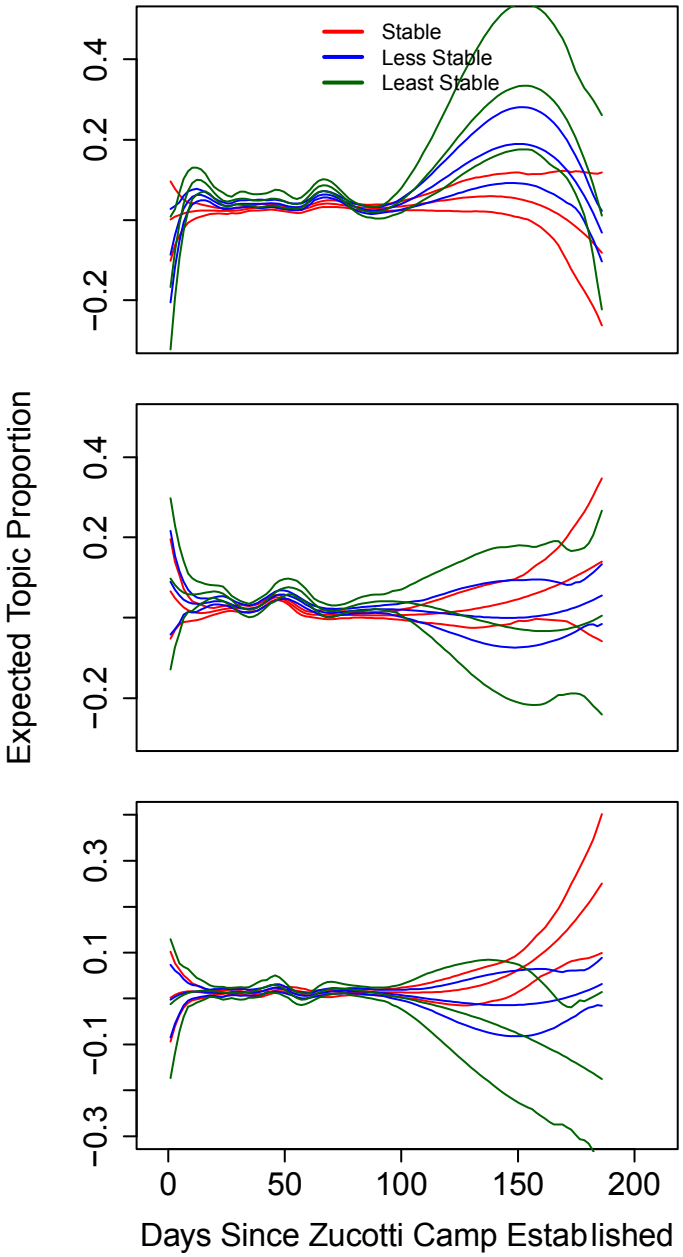
Figure X.5.18 – Rallies, Demonstrations, and Labor Alliances by Political Instability Interacted with Time



Note: Panel 1 (Top), Panel 2 (Middle), and Panel 3 (Bottom) show the predicted prevalence of Rallies, Demonstrations, and Labor Alliances in the corpus for each day

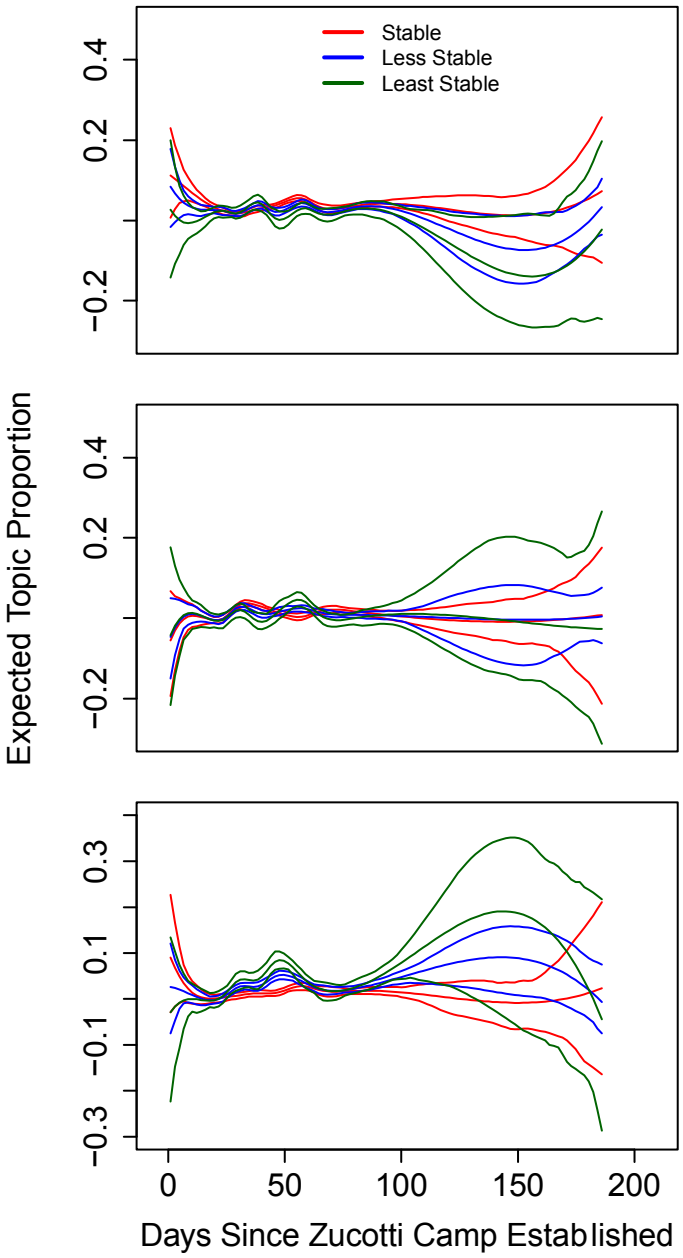
of the Occupy movement. Each panel displays three lines predicting topic prevalence for the highest, lowest, and middle value of the 'political instability' variable interacted with time. See Methodological Appendix for equations used in estimation and prediction.

Figure X.5.19– City Hall Targeting, Bank Targeting, and Sidewalk Contestation by Political Instability Interacted with Time



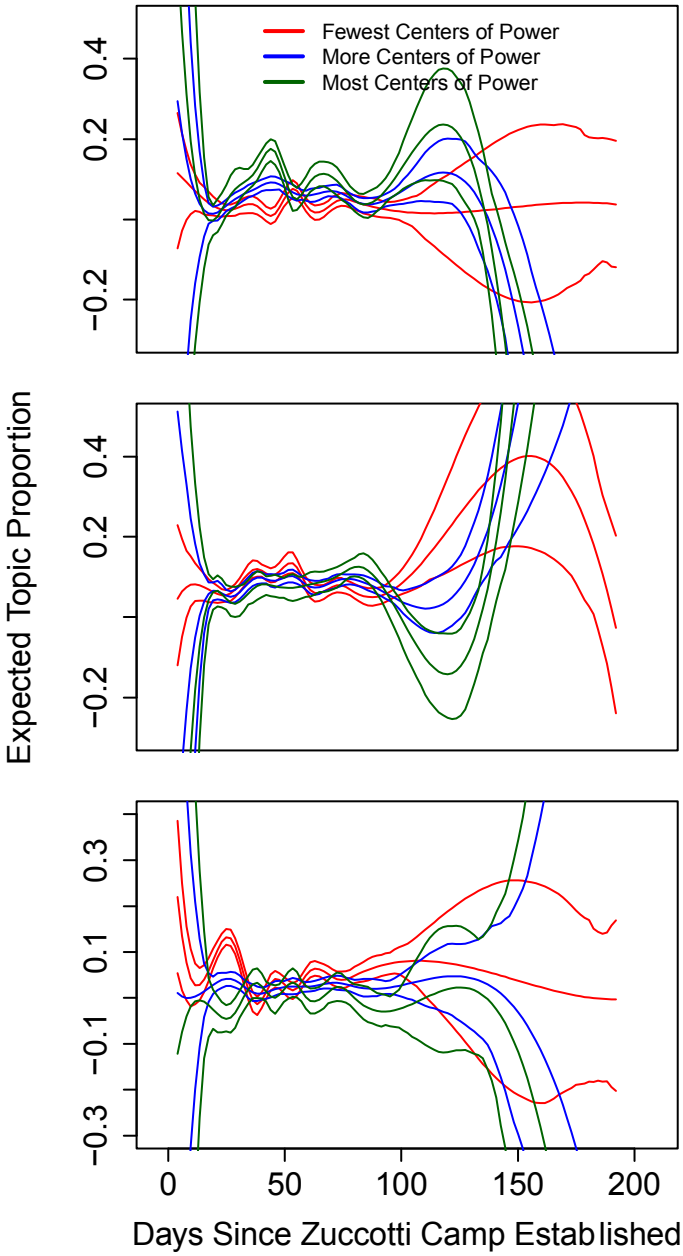
Note: Panel 1 (Top), Panel 2 (Middle), and Panel 3 (Bottom) show the predicted prevalence of City Hall Targeting, Bank Targeting, and Sidewalk Contestation in the corpus for each day of the Occupy movement. Each panel displays three lines predicting topic prevalence for the highest, lowest, and middle value of the 'political instability' variable interacted with time. See Methodological Appendix for equations used in estimation and prediction.

Figure X.5.20 – Traffic Battles, Curfew Disputes, and Arrests by Political Instability Interacted with Time



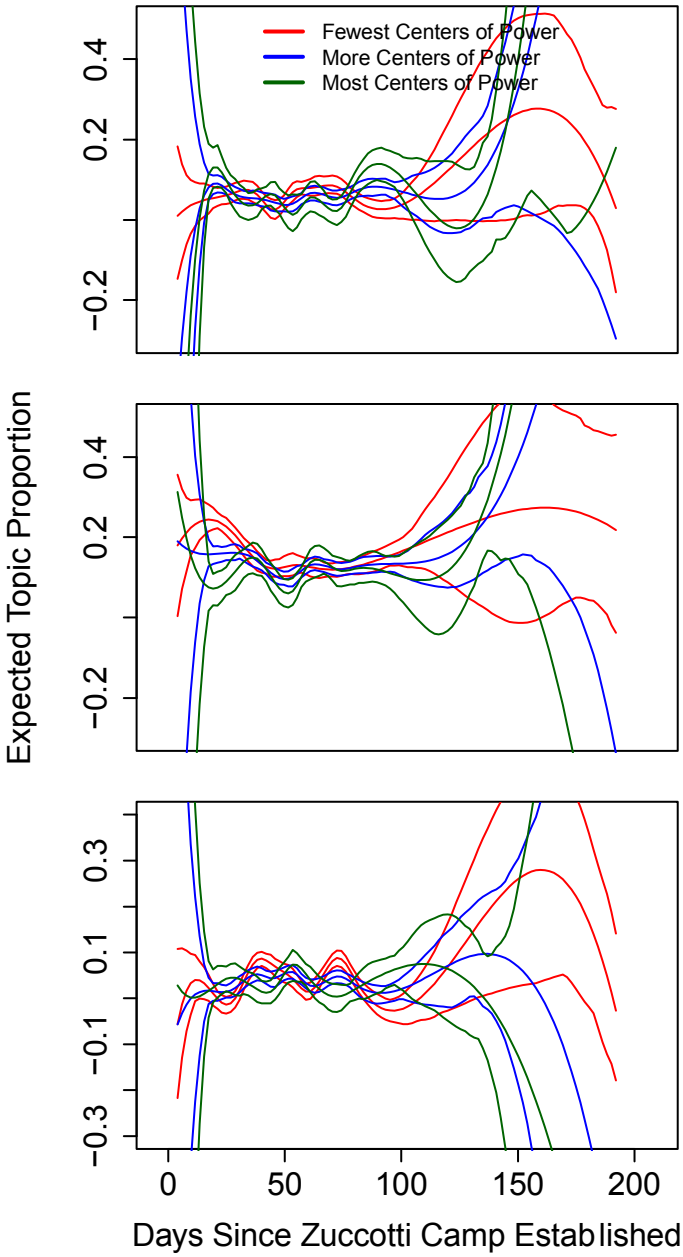
Note: Panel 1 (Top), Panel 2 (Middle), and Panel 3 (Bottom) show the predicted prevalence of Traffic Battles, Curfew Disputes, and Arrests in the corpus for each day of the Occupy movement. Each panel displays three lines predicting topic prevalence for the highest, lowest, and middle value of the 'political instability' variable interacted with time. See Methodological Appendix for equations used in estimation and prediction.

Figure X.6.1 – Ordinance Enforcing, Group Arresting, Arresting Resisting Individuals by Number of Power Centers Interacted with Time



Note: Panel 1 (Top), Panel 2 (Middle), and Panel 3 (Bottom) show the predicted prevalence of Ordinance Enforcing, Group Arresting, Arresting Resisting Individuals in the corpus of text units describing police-initiated events for each day of the Occupy movement. Each panel displays three lines predicting control performance prevalence for the highest, lowest, and middle value of the 'number of power centers' variable interacted with time. See Methodological Appendix for equations used in estimation and prediction.

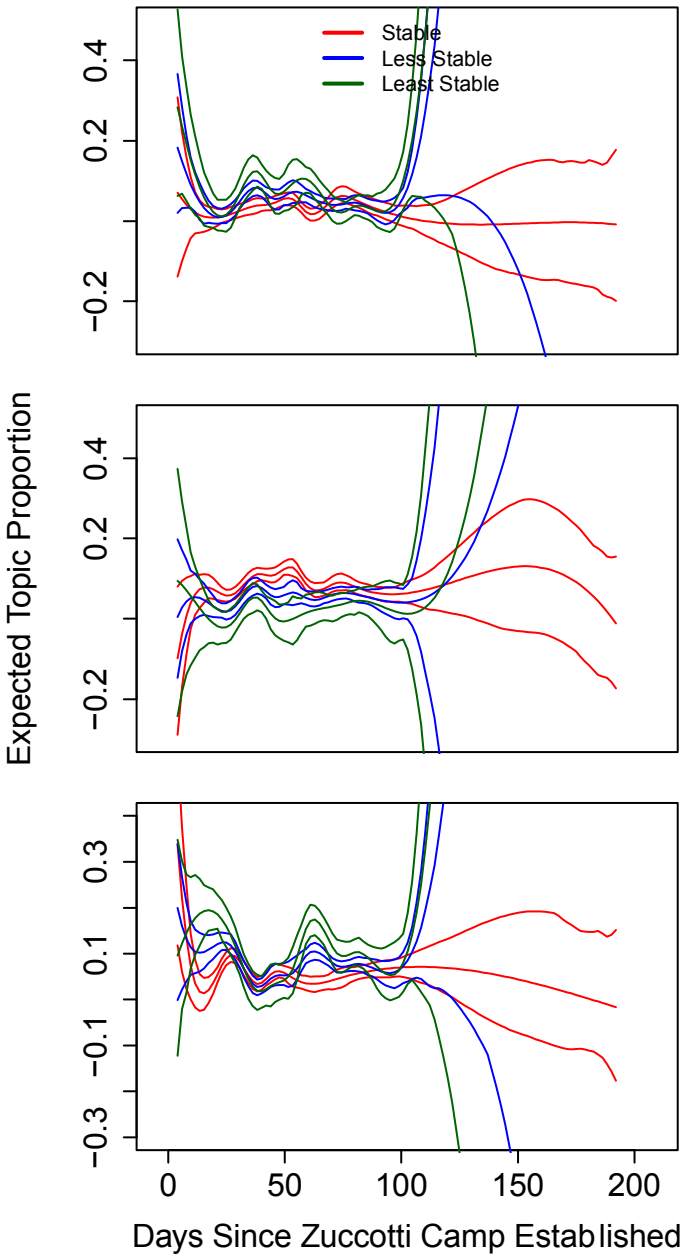
Figure X.6.2 – Deadline Enforcing, Dismantling Camps, and Violent Raiding by Number of Power Centers Interacted with Time



Note: Panel 1 (Top), Panel 2 (Middle), and Panel 3 (Bottom) show the predicted prevalence of Deadline Enforcing, Dismantling Camps, and Violent Raiding in the corpus of text units describing police-initiated events for each day of the Occupy movement. Each panel displays three lines predicting control performance prevalence for the highest, lowest, and middle value of the 'number of power centers' variable

interacted with time. See Methodological Appendix for equations used in estimation and prediction.

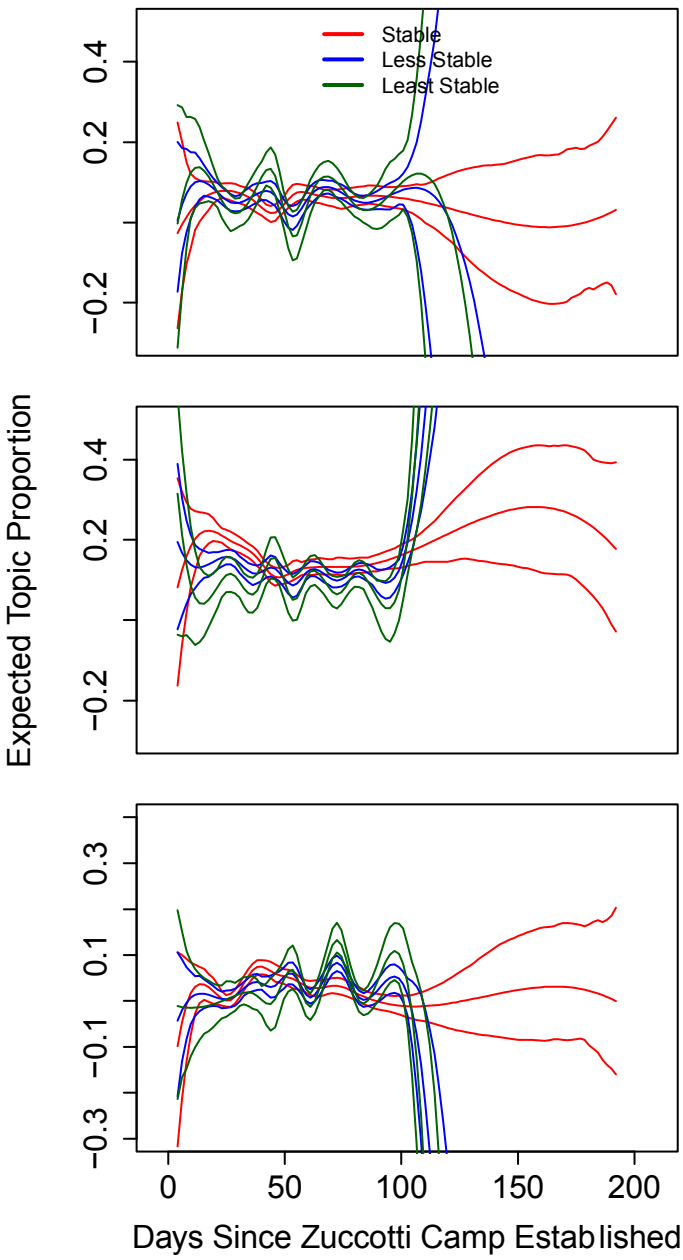
Figure X.6.3 – Ordinance Enforcing, Group Arresting, Arresting Resisting Individuals by Political Instability Interacted with Time



Note: Panel 1 (Top), Panel 2 (Middle), and Panel 3 (Bottom) show the predicted prevalence of Ordinance Enforcing, Group Arresting, Arresting Resisting Individuals in

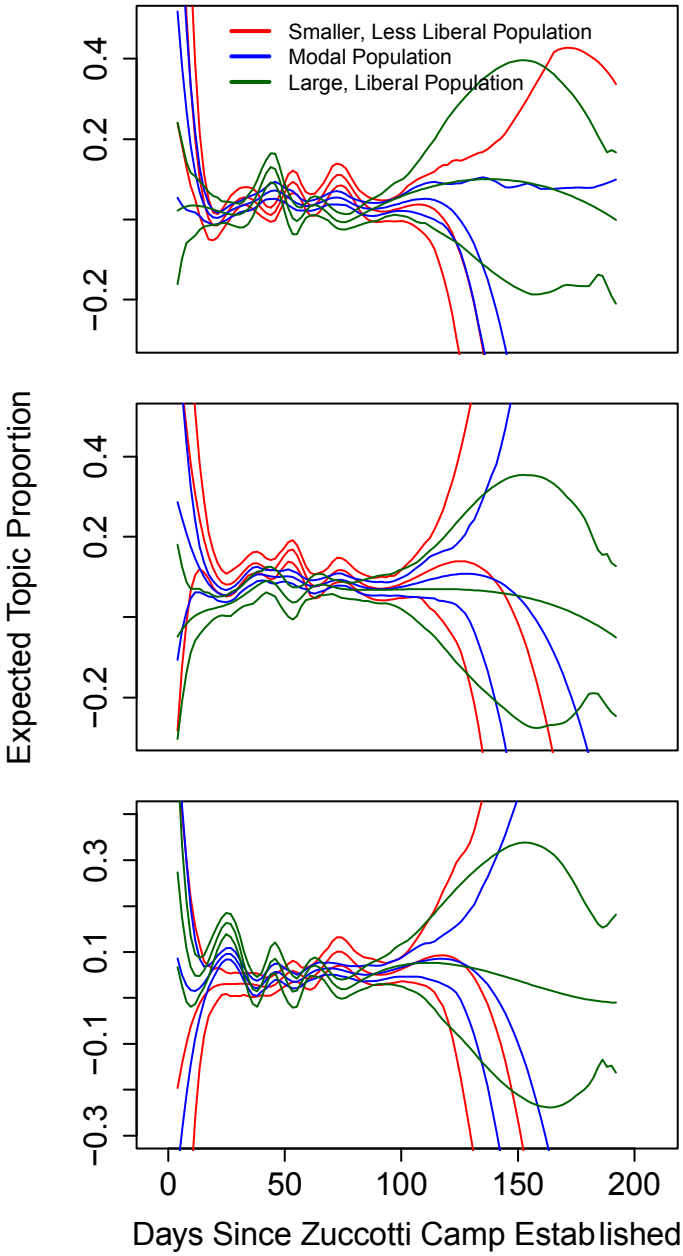
the corpus of text units describing police-initiated events for each day of the Occupy movement. Each panel displays three lines predicting control performance prevalence for the highest, lowest, and middle value of the 'instability' variable interacted with time. See Methodological Appendix for equations used in estimation and prediction.

Figure X.6.4 – Deadline Enforcing, Dismantling Camps, and Violent Raiding by Political Instability Interacted with Time



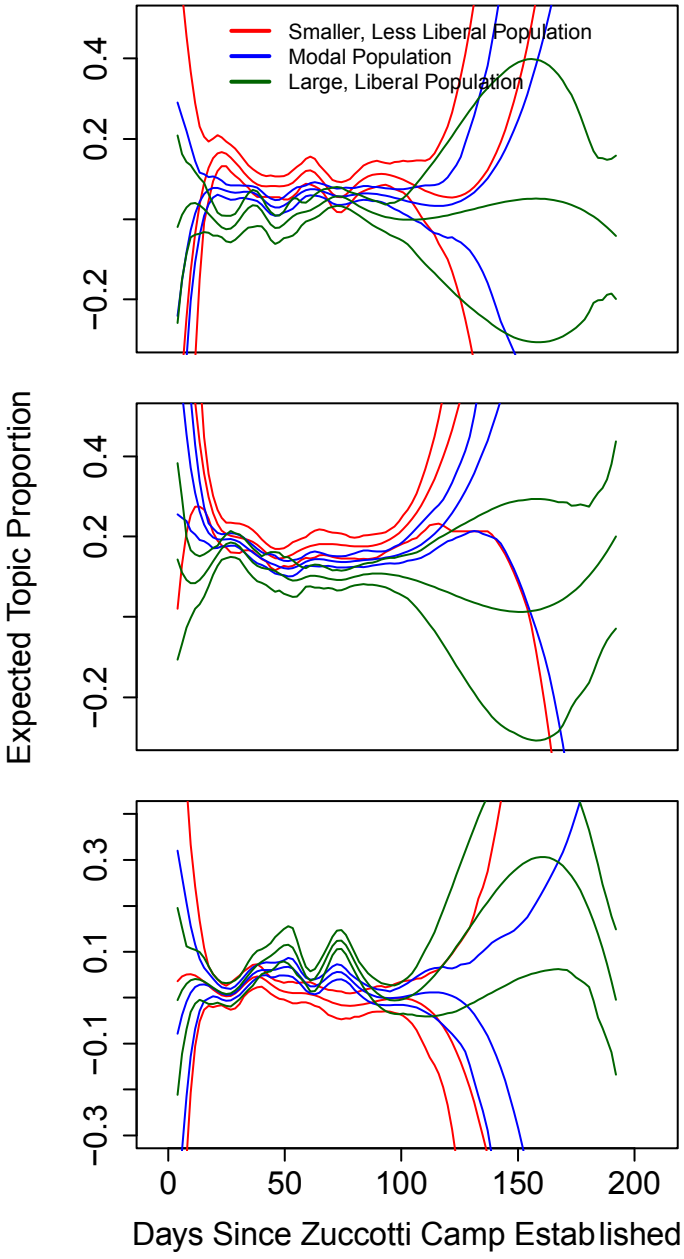
Note: Panel 1 (Top), Panel 2 (Middle), and Panel 3 (Bottom) show the predicted prevalence of Deadline Enforcing, Dismantling Camps, and Violent Raiding in the corpus of text units describing police-initiated events for each day of the Occupy movement. Each panel displays three lines predicting control performance prevalence for the highest, lowest, and middle value of the 'political instability' variable interacted with time. See Methodological Appendix for equations used in estimation and prediction.

Figure X.6.5 – Ordinance Enforcing, Group Arresting, Arresting Resisting Individuals by Size of Liberal Population Interacted with Time



Note: Panel 1 (Top), Panel 2 (Middle), and Panel 3 (Bottom) show the predicted prevalence of Ordinance Enforcing, Group Arresting, Arresting Resisting Individuals in the corpus of text units describing police-initiated events for each day of the Occupy movement. Each panel displays three lines predicting control performance prevalence for the highest, lowest, and middle value of the 'number of liberals' variable interacted with time. See Methodological Appendix for equations used in estimation and prediction.

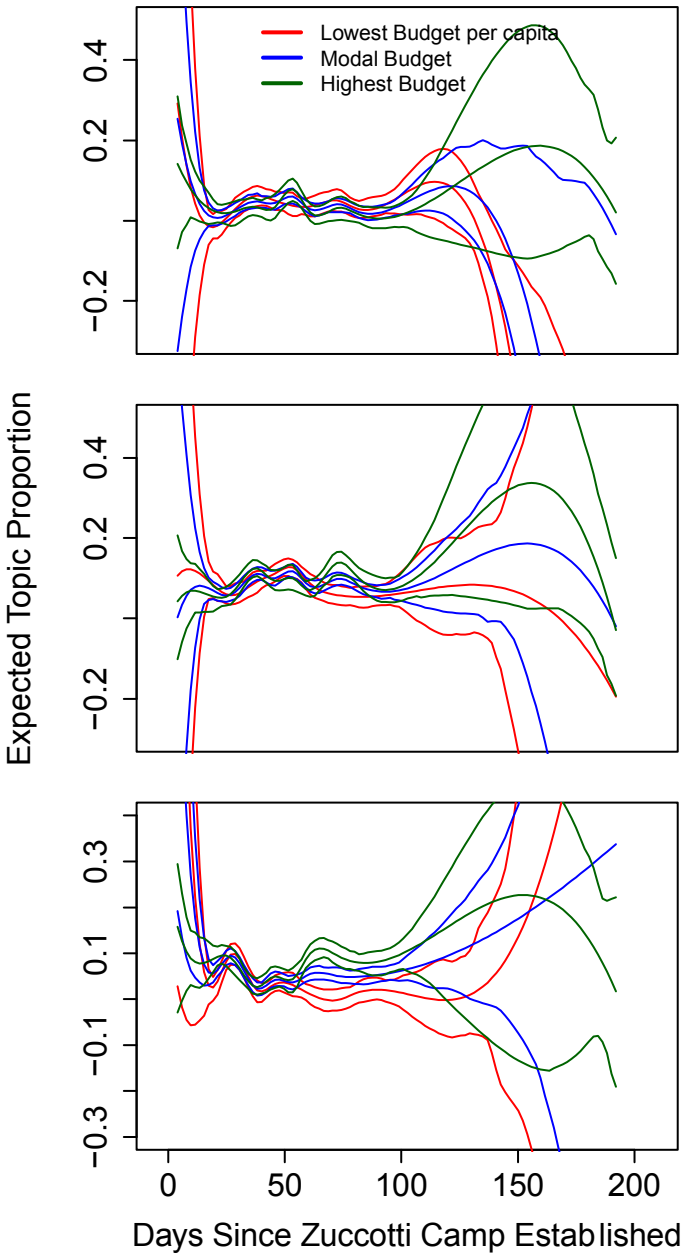
Figure X.6.6 – Deadline Enforcing, Dismantling Camps, and Violent Raiding by Size of Liberal Population Interacted with Time



Note: Panel 1 (Top), Panel 2 (Middle), and Panel 3 (Bottom) show the predicted prevalence of Ordinance Enforcing, Group Arresting, Arresting Resisting Individuals in the corpus of text units describing police-initiated events for each day of the Occupy movement. Each panel displays three lines predicting control performance prevalence for the highest, lowest, and

middle value of the 'number of liberals' variable interacted with time. See Methodological Appendix for equations used in estimation and prediction.

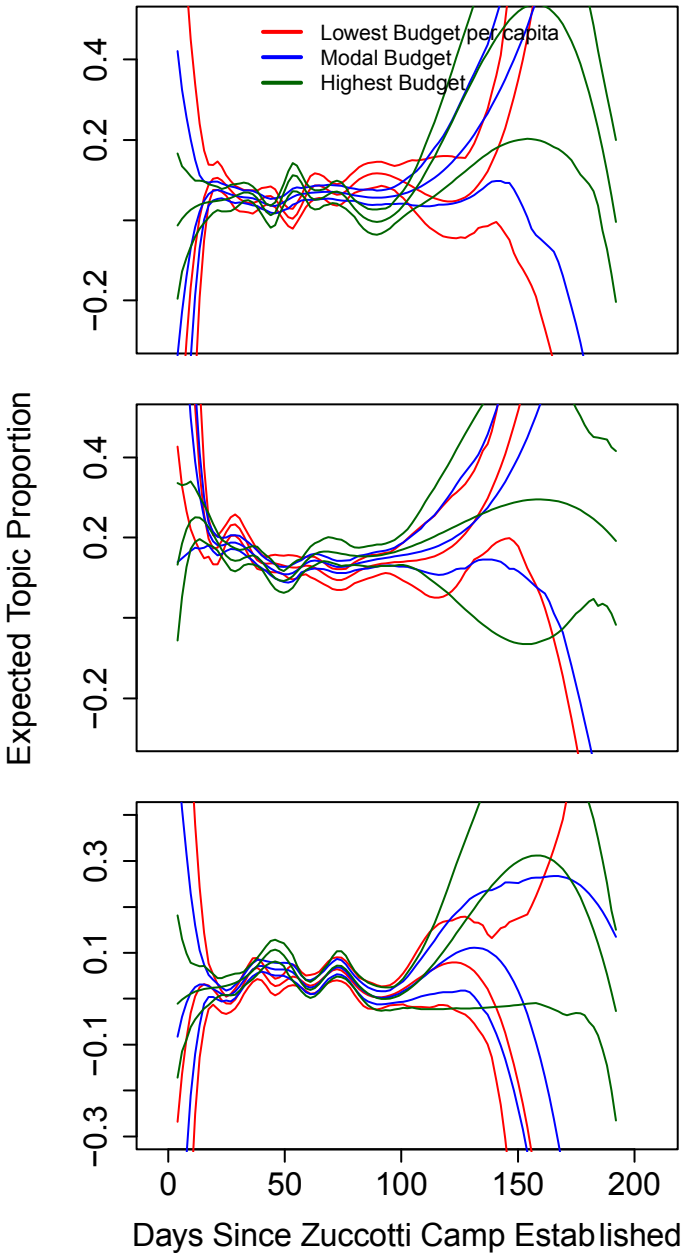
Figure X.6.7 – Ordinance Enforcing, Group Arresting, Arresting Resisting Individuals by Police Budget Interacted with Time



Note: Panel 1 (Top), Panel 2 (Middle), and Panel 3 (Bottom) show the predicted prevalence of Ordinance Enforcing, Group Arresting, Arresting Resisting Individuals in the corpus of text

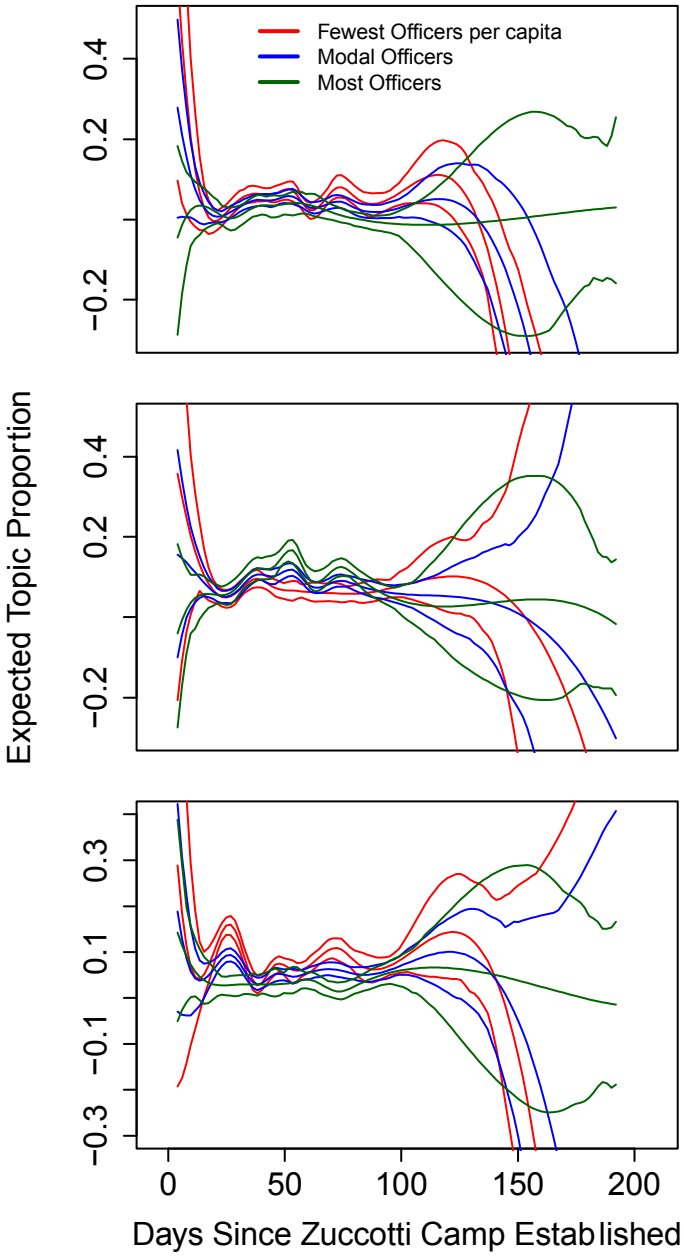
units describing police-initiated events for each day of the Occupy movement. Each panel displays three lines predicting control performance prevalence for the highest, lowest, and middle value of the 'police budget' variable interacted with time. See Methodological Appendix for equations used in estimation and prediction.

Figure X.6.8 – Deadline Enforcing, Dismantling Camps, and Violent Raiding by Police Budget Interacted with Time



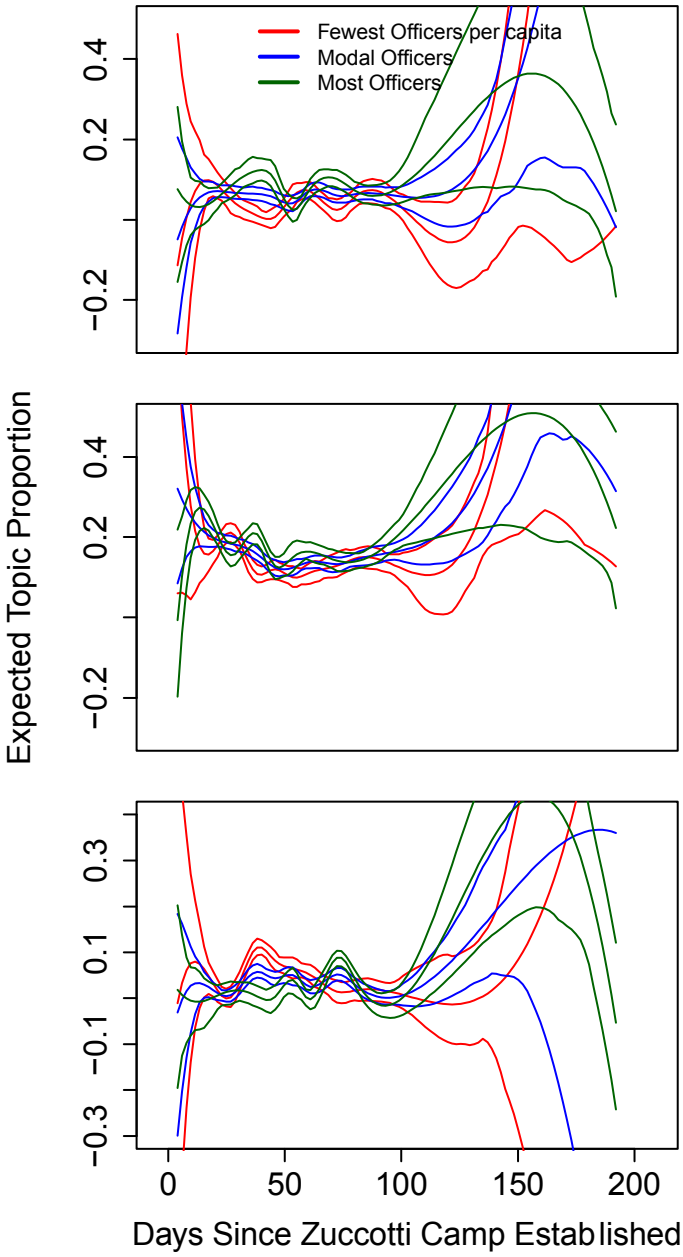
Note: Panel 1 (Top), Panel 2 (Middle), and Panel 3 (Bottom) show the predicted prevalence of Ordinance Enforcing, Group Arresting, Arresting Resisting Individuals in the corpus of text units describing police-initiated events for each day of the Occupy movement. Each panel displays three lines predicting control performance prevalence for the highest, lowest, and middle value of the 'police budget' variable interacted with time. See Methodological Appendix for equations used in estimation and prediction.

Figure X.6.9 – Ordinance Enforcing, Group Arresting, Arresting Resisting Individuals by Officers per Capita Interacted with Time



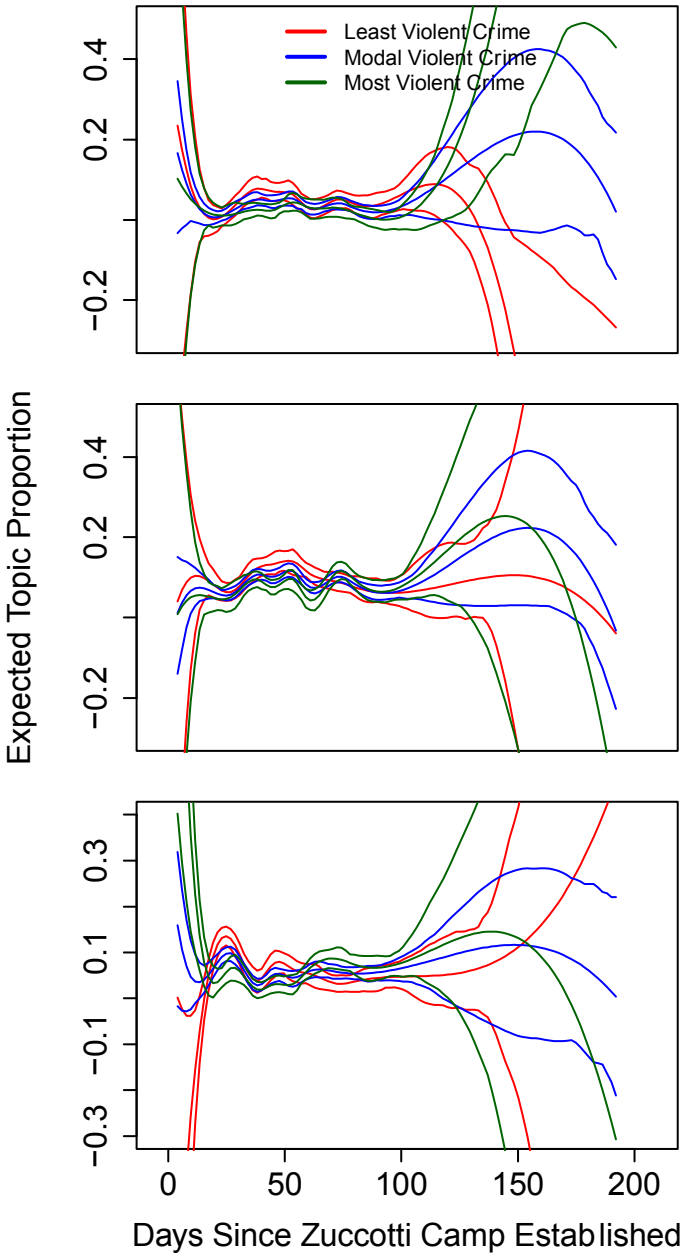
Note: Panel 1 (Top), Panel 2 (Middle), and Panel 3 (Bottom) show the predicted prevalence of Ordinance Enforcing, Group Arresting, Arresting Resisting Individuals in the corpus of text units describing police-initiated events for each day of the Occupy movement. Each panel displays three lines predicting control performance prevalence for the highest, lowest, and middle value of the 'officers per capita' variable interacted with time. See Methodological Appendix for equations used in estimation and prediction.

Figure X.6.10 – Deadline Enforcing, Dismantling Camps, and Violent Raiding by Officers per Capita Interacted with Time



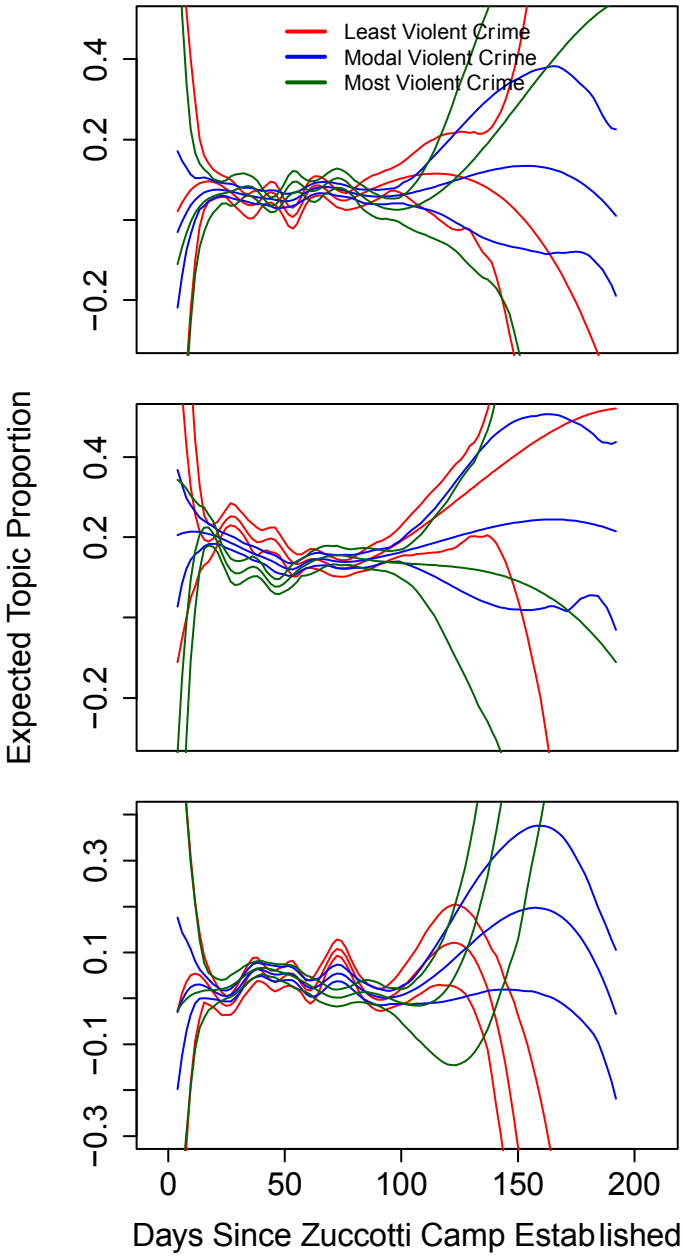
Note: Panel 1 (Top), Panel 2 (Middle), and Panel 3 (Bottom) show the predicted prevalence of Ordinance Enforcing, Group Arresting, Arresting Resisting Individuals in the corpus of text units describing police-initiated events for each day of the Occupy movement. Each panel displays three lines predicting control performance prevalence for the highest, lowest, and middle value of the 'officers per capita' variable interacted with time. See Methodological Appendix for equations used in estimation and prediction.

Figure X.6.11 – Ordinance Enforcing, Group Arresting, Arresting Resisting Individuals by Police Experience with Violent Crime Interacted with Time



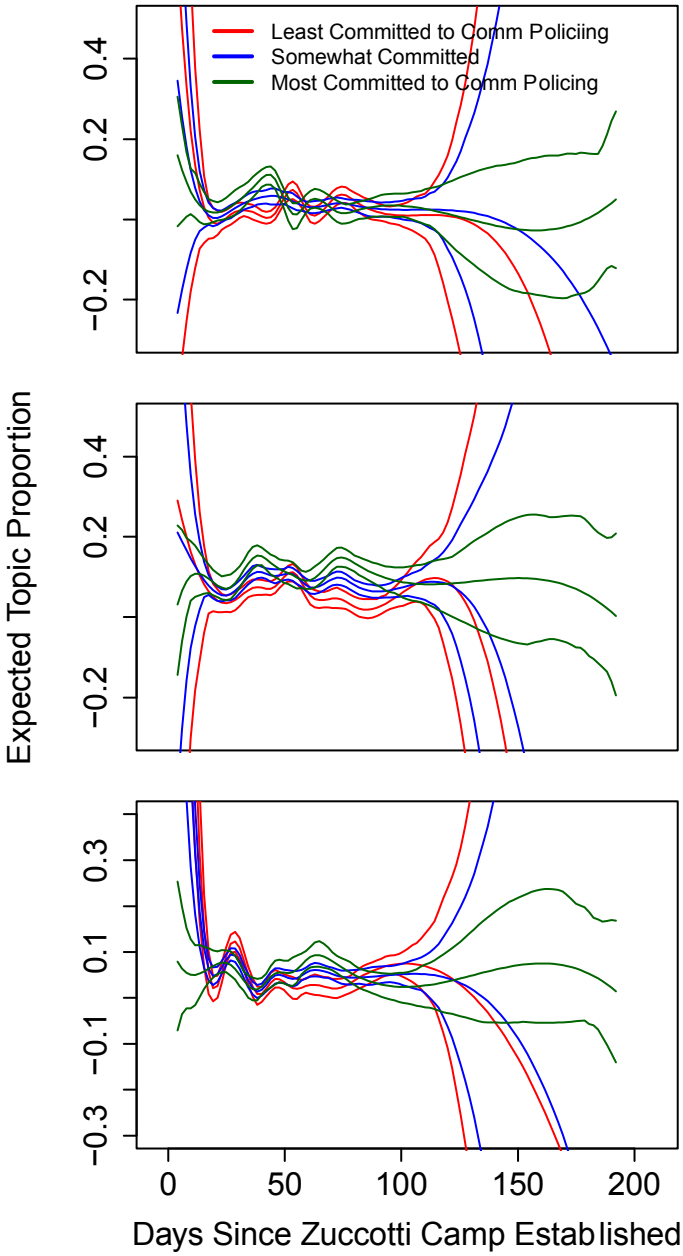
Note: Panel 1 (Top), Panel 2 (Middle), and Panel 3 (Bottom) show the predicted prevalence of Ordinance Enforcing, Group Arresting, Arresting Resisting Individuals in the corpus of text units describing police-initiated events for each day of the Occupy movement. Each panel displays three lines predicting control performance prevalence for the highest, lowest, and middle value of the 'police experience with violent crime' variable interacted with time. See Methodological Appendix for equations used in estimation and prediction.

Figure X.6.12 – Deadline Enforcing, Dismantling Camps, and Violent Raiding by Police Experience with Violent Crime Interacted with Time



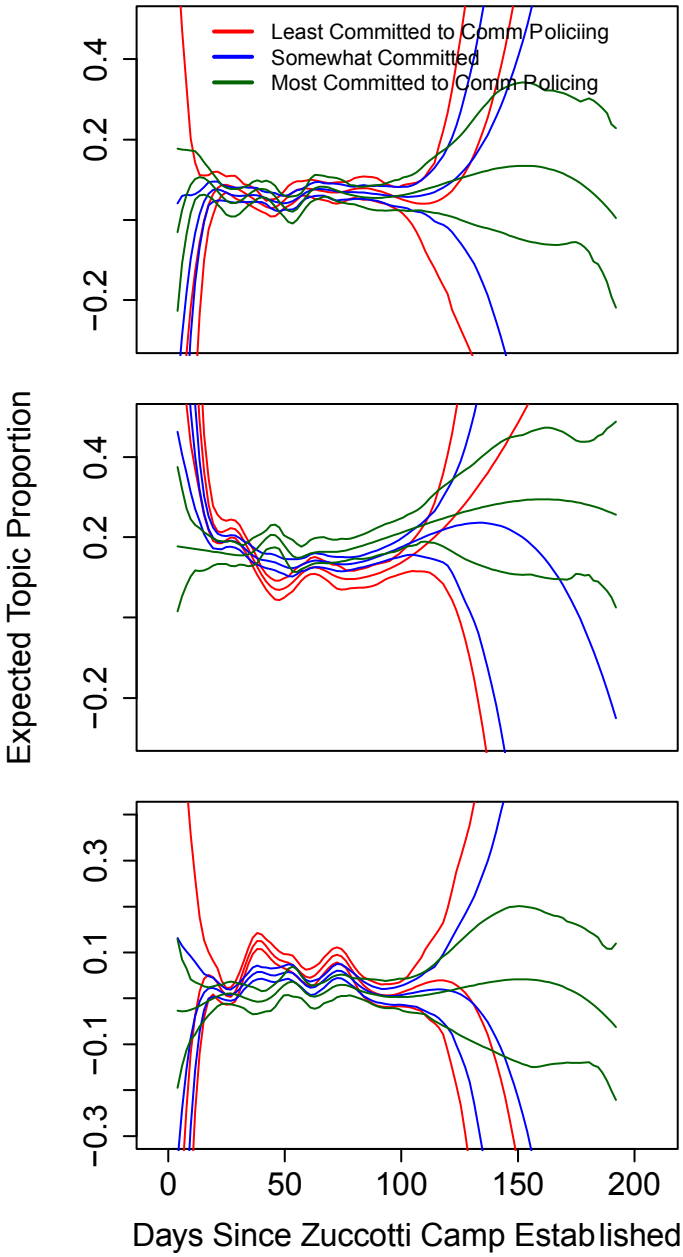
Note: Panel 1 (Top), Panel 2 (Middle), and Panel 3 (Bottom) show the predicted prevalence of Ordinance Enforcing, Group Arresting, Arresting Resisting Individuals in the corpus of text units describing police-initiated events for each day of the Occupy movement. Each panel displays three lines predicting control performance prevalence for the highest, lowest, and middle value of the 'police experience with violent crime' variable interacted with time. See Methodological Appendix for equations used in estimation and prediction.

Figure X.6.13 – Ordinance Enforcing, Group Arresting, Arresting Resisting Individuals by Commitment to Community Policing Interacted with Time



Note: Panel 1 (Top), Panel 2 (Middle), and Panel 3 (Bottom) show the predicted prevalence of Ordinance Enforcing, Group Arresting, Arresting Resisting Individuals in the corpus of text units describing police-initiated events for each day of the Occupy movement. Each panel displays three lines predicting control performance prevalence for the highest, lowest, and middle value of the 'community policing' variable interacted with time. See Methodological Appendix for equations used in estimation and prediction.

Figure X.6.14 – Deadline Enforcing, Dismantling Camps, and Violent Raiding by Commitment to Community Policing Interacted with Time



Note: Panel 1 (Top), Panel 2 (Middle), and Panel 3 (Bottom) show the predicted prevalence of Ordinance Enforcing, Group Arresting, Arresting Resisting Individuals in the corpus of text units describing police-initiated events for each day of the Occupy movement. Each panel displays three lines predicting control performance prevalence for the highest, lowest, and middle value of the 'community policing' variable interacted with time. See Methodological Appendix for equations used in estimation and prediction.