Title

Survival analysis of DNA mutation motifs with penalized proportional hazards

Journal

The Annals of Applied Statistics, 13(2)

ISSN

1932-6157

Authors

Feng, Jean
Shaw, David A
Minin, Vladimir N
et al.

Publication Date

2019-06-01

DOI

10.1214/18-aoas1233

Survival analysis of DNA mutation motifs with penalized proportional hazards

Jean Feng[§],

Department of Biostatistics, University of Washington
Seattle, WA, USA

David A. Shaw[§],

Computational Biology Program, Fred Hutchinson Cancer Research Center
Seattle, WA, USA

Vladimir N. Minin*,

Department of Statistics, University of California, Irvine
Irvine, CA, USA
e-mail: vminin@uci.org

Noah Simon*,

Department of Biostatistics, University of Washington
Seattle, WA, USA
e-mail: nrsimon@u.washington.edu

and

Frederick A. Matsen IV*,

Computational Biology Program, Fred Hutchinson Cancer Research Center
Seattle, WA, USA
e-mail: matsen@fredhutch.org

§ Co-first author
*** Co-corresponding authors**

Abstract: Antibodies, an essential part of our immune system, develop in an intricate process to guarantee a broad diversity of antibodies that are able to bind a continually diversifying array of pathogens. This process involves randomly mutating the DNA sequences that encode antibodies to find variants with improved binding. These mutations are not distributed uniformly across sequence sites. Immunologists observe this nonuniformity to be consistent with "mutation motifs", which are short DNA subsequences that affect how likely a given site is to experience a mutation. Quantifying the effect of motifs on mutation rates is challenging: a large number of possible motifs makes this statistical problem high dimensional, while the unobserved history of the mutation process leads to a nontrivial missing data problem. We introduce an ℓ_1-penalized proportional hazards model to infer mutation motifs and their effects. In order to estimate model parameters, our method uses a Monte Carlo EM algorithm to marginalize over

the unknown ordering of mutations. We show that our method performs better on simulated data compared to current methods and leads to more parsimonious models. The application of proportional hazards to analyses of mutation processes is, to our knowledge, novel and formalizes the current methods in a statistical framework that can be easily extended to analyze the effect of other biological features on mutation rates.

1. Introduction

We introduce a proportional hazards model approach to study DNA mutation processes. Our study is motivated by a mutation process that occurs in DNA sequences that encode B-cell receptors (BCRs), proteins that recognize and neutralize pathogens. BCRs are called antibodies in their secreted form. The immune system relies on this somatic hypermutation process to generate a diversity of BCRs that can bind to a large and continually evolving variety of pathogens. The mutation process begins with a naive BCR sequence that is formed by randomly recombining germline DNA sequences in a process called V(D)J recombination (Tonegawa, 1983; Schatz and Ji, 2011). Mutations are then introduced by the mutagenic enzyme activation-induced cytidine deaminase (AID) and error-prone DNA repair (Chahwan et al., 2012; Methot and Di Noia, 2017). This specialized machinery introduces mutations in a random yet nonuniform pattern and is known to be highly sensitive to the sequence motif: the sequence of DNA bases surrounding the mutating position. In fact, previous work has shown that there is over an order of magnitude difference in the mutability between nearby nucleotide positions (Yaari et al., 2013; Yeap et al., 2015; Cui et al., 2016). Immunologists have described these motifs for decades: Early work focuses on finding small sets of motifs that are especially mutable (Rogozin and Kolchanov, 1992; Dunn-Walters et al., 1998) while later work, more quantitatively, describes rates of mutation for all possible motifs of a given length (Pham et al., 2003; Yaari et al., 2013; Bonissone and Pevzner, 2015; Cui et al., 2016) or uses regression to model the influence of nearby bases on mutation rates (Cohen, Kleinstein and Louzoun, 2011; Elhanati et al., 2015).

Our goal is to develop a solid statistical framework that estimates the mutation rates of motifs and provides interpretable results for this mutation process. A better understanding of somatic hypermutation will help us understand how potent antibodies arise naturally and how to design vaccines for challenging viruses such as HIV, because the somatic mutation process determines which evolutionary paths are more or less likely for a given immunization (Haynes et al., 2012; Hwang et al., 2017). In addition, characterizing the mutation process is also essential to understanding the natural selection processes that refine the antibody repertoire (Hershberg et al., 2008; Uduman et al., 2011; McCoy et al., 2015; Hoehn, Lunter and Pybus, 2017). Finally, estimating the mutability of motifs paves the way to a more mechanistic understanding of the mutation process, such as understanding the relative role of mutation (Pham et al., 2003) and error-prone repair (Rogozin et al., 2001).

A number of strategies have been used to model the mutability of motifs. The

general approach is to compare a mutated sequence with an inferred naive sequence and model the differences between them. Cohen, Kleinstein and Louzoun (2011) and Elhanati et al. (2015) model the mutabilities of motifs as the product of the mutabilities of short subsequences (usually 1 or 2 bases). By using a log-linear model with only first-order terms, they prevent a combinatorial explosion in the number of parameters to estimate, but miss interactions between the positions. Yaari et al. (2013) and Cui et al. (2016) do not use such a log-linear assumption and instead estimate the mutability of all possible five-nucleotide motifs, using ad-hoc methods to handle motifs with few observations. Rather than these restrictive and ad-hoc approaches, a more data-adaptive variable selection method is desirable.

Another drawback of the aforementioned methods is that they ignore mutations that occur in neighboring positions, even though such events can carry important information about highly mutable motifs. The issue is that these methods require counting the number of times a motif is observed to mutate; if mutations occur in neighboring positions, they cannot attribute the mutation to the correct motif. For settings with high rates of mutation, these methods end up estimating the mutabilities poorly. To properly estimate mutabilities, one needs to account for the different possible orders that mutations occurred in. Previous work has developed methods for performing various types of inference when mutation order is unknown (Hwang and Green, 2004; Hobolth, 2008), but these inference procedures estimate time (or a scalar multiple of time) between observed nucleotide sequences, making a parametric assumption that the mutation process follows a continuous time Markov process. Here we are not interested in estimating time so we can relax the model assumptions.

In this paper, we advance the modeling of motif mutabilities in several directions. We propose a method to fit mutabilities using survival analysis of mutation motifs, called `samm`. We formalize the problem using Cox proportional hazards, in which mutations are the failure events to be investigated. Although survival models are used implicitly by computational immunologists for simulation (Yaari, Uduman and Kleinstein, 2012; Sheng et al., 2017), we believe this is the first time they have been used for inference.

To estimate motif mutabilities, our method uses the Monte Carlo expectation–maximization algorithm (MCEM, Wei and Tanner, 1990). Since the orders in which mutations occur are unobserved in our data, expectation–maximization (EM, Dempster, Laird and Rubin, 1977) allows us to perform maximum likelihood while averaging over these unknown orders. However, this averaging requires calculating the expected log-likelihood, which is analytically intractable in our case; we estimate the expectation using Gibbs sampling. This approach is similar to that used by Goggins et al. (1998) to model interval-censored failure-time data where the order in which the failure events occur is unknown.

Our method also handles high-dimensional settings where there are many more predictors than observations, which is important because many motifs are hypothesized to affect the mutation rate but the specific ones are unknown. For instance, Yaari et al. (2013) and Cui et al. (2016) consider all motifs of length 5. We use the lasso (Tibshirani, 1996) to improve estimation and perform vari-

able selection. To provide a measure of uncertainty of our estimates, we use a two-step approach: we fit an ℓ_1-penalized Cox proportional hazards model (Tibshirani et al., 1997) to perform variable selection and refit an unpenalized model over the selected variables to obtain our final estimates along with approximate confidence intervals.

The paper is organized as follows. Section 2 describes our two-step estimation method. Section 3 presents simulation results. In Section 4, we apply our method to model somatic hypermutation of BCR sequences from Cui et al. (2016).

2. Methods

Our data consists of BCR nucleotide sequences that have mutated for an unknown period of time. Though we focus on modeling the somatic hypermutation process of BCRs, our approach can be framed more generally as a problem of modeling a mutation process of sequence-valued data. We refer to the original, unmutated sequences as "naive" and their descendants as "mutated" sequences. Throughout, we suppose that these naive sequences are known. In the BCR case, we restrict our attention to a computationally-identified naive segment coded in germline DNA (the V region, Yaari and Kleinstein, 2015).

More formally, the mutation process of a sequence with p positions can be described as a vector-valued stochastic process $\{X(t) = (X_1(t), ..., X_p(t)) : t \in [0, \infty)\}$ indexed by time t. Each $\{X_j(t)\}$ represents the mutation process of the jth position in the sequence. For a given time t, the state space of $X_j(t)$ is the set of nucleotides $\{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$ and the state space of $X(t)$ is the set of nucleotide sequences $\mathcal{S} = \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}^p$. At the start of the mutation process, $X(0)$ is fixed to be the naive sequence.

We use a survival analysis framework to model the mutation process. We view positions in a single sequence as subjects observed for the same time period. A mutation event at position j occurs at time t if the nucleotide immediately before time t, $\lim_{s \to t^-} X_j(s)$, differs from the nucleotide at time t, $X_j(t)$. If a position never mutates, we consider its mutation time to be censored. Each position is assumed to evolve independently, but with a per-site process that depends on the state of other positions in the sequence.

The hazard (or mutation) rate of a position is the instantaneous risk of mutating at time t given that it has been conserved up to time t. In a context-sensitive model, the hazard rate at each position at time t depends on the current nucleotide sequence $X(t)$. When a mutation event occurs along the sequence, the mutation rate of each position may change (Figure 1). In our work, we assume that only local context matters: A mutation event at some position will only affect the mutation rate of some range of neighboring positions. Since we do not observe the order of mutation events in the data and only observe pairs of naive and mutated sequences, there are many possible mutation orders that could explain how the mutated sequence arose from the naive sequence; each mutation order corresponds to a distinct sequence of hazard rates.

For ease of exposition, we present our estimation method for the mutation process of a single pair of naive and mutated nucleotide sequences. The method

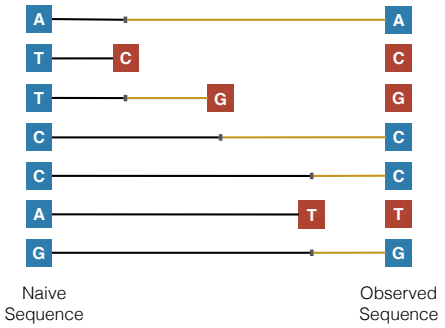


FIG 1. *Survival analysis for BCR sequences. In a context-dependent mutation model, a mutation event can change the mutation rates of other positions. Suppose the hazard (i.e. mutation) rate of a position depends on the position's two neighboring bases. Then, for example, when the* `T` *in the third position mutates to a* `G`*, the hazard rate for* `C` *in the fourth position changes from the original* `TCC` *motif to the* `GCC` *motif. Changes in the hazard rate are indicated by a change in line color from black to yellow.*

readily applies to estimating rates given many independent mutation processes (a typical application will be to thousands or more sequences). We assume that each position can mutate at most once, called parsimony or the parsimony principle in evolutionary biology.

2.1. Cox proportional hazards

We now detail the hazard rate model. A motif m is a sequence of nucleotides with length $\text{len}(m)$, where $\text{len}(m)$ is typically much smaller than the number of nucleotides in $X(t)$. The function $I(X(t), m, j, j')$ indicates if motif m appears in sequence $X(t)$ from positions $j - j' + 1$ to $j - j' + \text{len}(m)$ by evaluating

$$I(X(t), m, j, j') = \prod_{k=1}^{\text{len}(m)} 1\left\{X_{j-j'+k}(t) = m_k\right\}, \tag{1}$$

where m_k is the nucleotide in the kth position of motif m. In the special case where m is odd and $j' = (\text{len}(m) + 1)/2$, (1) checks if motif m is centered at position j in $X(t)$. This is called a centered motif; for all other cases we say that (1) is checking for an offset motif.

Define a motif dictionary to be a set $\mathcal{M}$ of sequence features (m, j') that may affect mutation rate. Dictionaries we consider include: 1-mers (length 1 motifs), offset 2-mers (length 2 motifs with $j' = 1, 2$), all of the central and offset 3-mers (length 3 motifs, with $j' = 1, 2, 3$), and all of the central 5-mers. We additionally consider all possible unions of these dictionaries. To ease exposition, we fix $\mathcal{M}$ to be $\left\{(m^{(1)}, j'^{(1)}), \ldots, (m^{(q)}, j'^{(q)})\right\}$ with an arbitrarily assigned but fixed order. We may now define a function that indicates which motifs in $\mathcal{M}$ occur at each position. For each position j, let $\psi_j : \mathcal{S} \mapsto \{0, 1\}^q$ be defined by

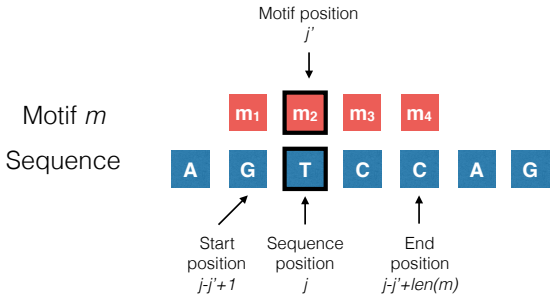


FIG 2. *An example of how feature vectors are generated: if we believe that the mutation rate at a position depends on the 4-mer motif starting one position to its left, then the feature vector for position j is a one-hot encoding of the sequence that appears in position $j-1$ through $j+2$. More formally, each element in the feature vector at position j indicates whether or not a motif m appears from start position $j-j'+1$ through end position $j-j'+\text{len}(m)$ (here $m=4$ and $j'=2$). The start and end positions are derived by aligning position j of the sequence with position j' of the motif.*

$[\psi_j(X(t))]_k \equiv I(X(t), m^{(k)}, j, j'^{(k)})$. We use ψ_j as the feature vector for modeling the mutation rate of position j (Figure 2).

The mutation rate of position j at time t is modeled as

$$h_j(t) = h_0(t) \exp\left(\boldsymbol{\theta}^\top \psi_j\left(X(t)\right)\right), \tag{2}$$

where $\boldsymbol{\theta} \in \mathbb{R}^q$ and the baseline hazard rate $h_0(\cdot)$ are unknown. Extending (2), we can additionally model the rate at which our process mutates to a specific nucleotide – the target nucleotide. Previous work (Cowell and Kepler, 2000; Rogozin et al., 2001; Yaari et al., 2013; Cui et al., 2016) suggests that the context-dependent mutation process is biased in favor of mutations to particular bases. We can take into account these preferences by additionally defining vectors $\boldsymbol{\theta}_{\mathtt{N}}$ for each possible target nucleotide $\mathtt{N} \in \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$. Using a competing events framework, the rate of mutating to nucleotide $\mathtt{N}$ at position j at time t is modeled as

$$h_{j\to\mathtt{N}}(t) = 1\{X_j(t) \neq \mathtt{N}\} h_0(t) \exp\left(\left(\boldsymbol{\theta} + \boldsymbol{\theta}_{\mathtt{N}}\right)^\top \psi_j(X(t))\right). \tag{3}$$

The indicator function $1\{\cdot\}$ in (3) ensures that positions cannot mutate to the same nucleotide. Section 2.3 will clarify why we have both $\boldsymbol{\theta}$ and $\boldsymbol{\theta}_{\mathtt{N}}$ in (3).

2.2. Maximum likelihood via MCEM

We are now ready to present a maximum likelihood estimation method for our model. We assume that the hazard rate follows (2). The per-target model in (3) is a straightforward extension of this simpler case. Let the observed data, namely the single pair of naive and mutated nucleotide sequences, be denoted $\mathbf{S}_{\text{obs}}$. Suppose n positions have mutated and we observe the mutation order;

use π_j to denote the location on the DNA sequence of the jth mutation in the order.

When $h_0(t)$ is completely unknown, only the order of the mutations carry information about $\boldsymbol{\theta}$, even if the mutation times are observed (Kalbfleisch and Prentice, 2011). An intuitive explanation is that without information on h_0, time can be transformed by an arbitrary increasing function and the form of the hazard function would still be of the form (2). (For more details, see Chapter 4 in Kalbfleisch and Prentice, 2011). Consequently, estimating $\boldsymbol{\theta}$ only involves maximizing the likelihood of observing the mutation order.

Define $\boldsymbol{\pi}_{1:j}$ to be the positions of the first through jth mutation, where $\boldsymbol{\pi}_{1:0}$ is defined to be the empty set. Define $S(\boldsymbol{\pi}_{1:j})$ to be the nucleotide sequence after positions $\boldsymbol{\pi}_{1:j}$ mutate. By the parsimony assumption, $R(\boldsymbol{\pi}_{1:j}) \equiv \{1, ..., p\} \setminus \boldsymbol{\pi}_{1:j}$ is the set of positions at risk of mutating, commonly referred to as the risk group in the survival analysis literature. Then the marginal likelihood of $\boldsymbol{\theta}$ is

$$\mathcal{L}_c(\mathbf{S}_{\text{obs}}, \boldsymbol{\pi}; \boldsymbol{\theta}) = \prod_{j=1}^{n} \frac{\exp(\boldsymbol{\theta}^\top \psi_{\pi_j}(S(\boldsymbol{\pi}_{1:j-1})))}{\sum_{k \in R(\boldsymbol{\pi}_{1:j-1})} \exp(\boldsymbol{\theta}^\top \psi_k(S(\boldsymbol{\pi}_{1:j-1})))}. \tag{4}$$

Our result looks like the marginal likelihood derived in equation 4.47 in Kalbfleisch and Prentice (2011) except that it is derived under a more general set of assumptions – whereas they assume the covariates are fixed, we assume the covariates to be fixed between events. The derivation of (4) is given in the Appendix.

The marginal likelihood in (4) implies that the mutation order can be simulated by drawing positions from successive multinomial distributions. To determine the jth position to mutate, we draw a position from the risk group $R(\boldsymbol{\pi}_{1:j-1})$. In fact, Gupta et al. (2015) use this procedure to simulate the somatic hypermutation process, though they do not provide a statistical justification. The marginal likelihood in (4) shows that simulating mutations using successive multinomial draws follows a Cox proportional hazards model with no assumptions on the baseline hazard rate.

Unfortunately the mutation order $\boldsymbol{\pi}$ is unknown in our problem. We instead maximize the observed data likelihood, which is the complete data likelihood marginalized over all admissible mutation orders $\mathcal{A}(\mathbf{S}_{\text{obs}})$:

$$\mathcal{L}(\mathbf{S}_{\text{obs}}; \boldsymbol{\theta}) = \sum_{\boldsymbol{\pi} \in \mathcal{A}(\mathbf{S}_{\text{obs}})} \mathcal{L}_c(\mathbf{S}_{\text{obs}}, \boldsymbol{\pi}; \boldsymbol{\theta}). \tag{5}$$

Under the parsimony assumption, $\mathcal{A}(\mathbf{S}_{\text{obs}})$ is a set of $n!$ possible mutation orders. When the number of mutated positions n is small, we can enumerate all possible mutation orders and maximize (5) using a nonlinear optimization algorithm such as EM (Dempster, Laird and Rubin, 1977). However, in most data sets n is much too large for direct enumeration to be computationally tractable, so we maximize (5) using MCEM.

MCEM extends the traditional EM algorithm by approximating the expectation in the E-step using a Monte Carlo sampling method. Let $\boldsymbol{\pi} = \boldsymbol{\pi}_{1:n}$ be a full mutation order. In this paper, we use the Gibbs sampler in Algorithm 1

to sample full mutation orders from the stationary distribution $\boldsymbol{\pi} \mid \{\mathbf{S}_{\text{obs}}, \boldsymbol{\theta}\}$. Given a full mutation order $\boldsymbol{\pi}$, let $\boldsymbol{\pi}_{(-j)}$ be the partial mutation order where the j-th mutation is removed from $\boldsymbol{\pi}$. For instance, if $\boldsymbol{\pi} = [1, 3, 2]$ then the partial mutation order $\boldsymbol{\pi}_{(-2)}$ is $[1, 2]$. At iteration k of Algorithm 1, we sample a full mutation order $\boldsymbol{\pi}^{(k)}$ that is consistent with the partial mutation order $\boldsymbol{\pi}_{(-j)}^{(k-1)}$ for a position $j \in \{1, ..., n\}$. The proof that this sampler converges to the desired probability distribution is standard and similar to that of Goggins et al. (1998).

We efficiently calculate the probability of a full mutation order given a partial mutation order by reusing previous computations. Let $\boldsymbol{\pi}'$ and $\boldsymbol{\pi}''$ be full mutation orders consistent with $\boldsymbol{\pi}_{(-j)}$, where $\boldsymbol{\pi}'$ and $\boldsymbol{\pi}''$ have the same mutation order except the jth and $(j+1)$th mutation are swapped. The ratio of the conditional probabilities of $\boldsymbol{\pi}'$ and $\boldsymbol{\pi}''$ given $\boldsymbol{\pi}_{(-j)}$ is

$$\frac{\Pr(\boldsymbol{\pi}'|\boldsymbol{\pi}_{(-j)})}{\Pr(\boldsymbol{\pi}''|\boldsymbol{\pi}_{(-j)})} = \frac{\exp\left(\boldsymbol{\theta}^\top \left(\psi_{\pi'_j}\left(S(\boldsymbol{\pi}'_{1:j-1})\right) + \psi_{\pi'_{j+1}}\left(S(\boldsymbol{\pi}'_{1:j})\right)\right)\right)}{\exp\left(\boldsymbol{\theta}^\top \left(\psi_{\pi''_j}\left(S(\boldsymbol{\pi}_{1:j-1})\right) + \psi_{\pi''_{j+1}}\left(S(\boldsymbol{\pi}_{1:j})\right)\right)\right)} \times \frac{\sum_{i \in R(\boldsymbol{\pi}''_{1:j})} \exp\left(\boldsymbol{\theta}^\top \psi_i\left(S(\boldsymbol{\pi}''_{1:j})\right)\right)}{\sum_{i \in R(\boldsymbol{\pi}'_{1:j})} \exp\left(\boldsymbol{\theta}^\top \psi_i\left(S(\boldsymbol{\pi}'_{1:j})\right)\right)}. \tag{6}$$

So if we already have $\Pr(\boldsymbol{\pi}''|\boldsymbol{\pi}_{(-j)})$, we can multiply it by (6) to quickly obtain $\Pr(\boldsymbol{\pi}'|\boldsymbol{\pi}_{(-j)})$. Moreover, we can efficiently calculate (6) by storing previous computational results: for instance, the summation over the risk group $R(\boldsymbol{\pi}'_{1:j})$ shares many elements with the summation over the risk group $R(\boldsymbol{\pi}''_{1:j})$. Similar ideas can be used to speed up other calculations required for MCEM.

Algorithm 1 Gibbs sampler for mutation orders

```
Initialize k = 1 and π^(0).
for iteration i = 1, 2, ... do
  for position j ∈ {1, ..., n} do
    π_(−j) := π_(−j)^(k−1)
    Sample π^(k) from the distribution Pr(π | π_(−j)^(k−1)).
    k := k + 1
  end for
end for
```

Given the Monte Carlo samples from the E-step, the M-step maximizes the mean log-likelihood of the complete data. Suppose the E-step generates Monte Carlo samples $\boldsymbol{\pi}^{(1)}, ..., \boldsymbol{\pi}^{(E)}$. Then during the M-step, we solve

$$\max_{\boldsymbol{\theta}} \frac{1}{E} \sum_{i=1}^{E} \log \mathcal{L}_c\left(\mathbf{S}_{\text{obs}}, \boldsymbol{\pi}^{(i)}; \boldsymbol{\theta}\right) \tag{7}$$

using iterative procedures such as gradient ascent.

We use ascent-based MCEM (Caffo, Jank and Jones, 2005) to maintain the monotonicity property of the EM algorithm. Briefly, ascent-based MCEM gives

a rule for deciding if the proposed MCEM estimate at each iteration should be accepted or if the Monte Carlo sample size should be increased. As the number of Monte Carlo samples increases, the standard error of the estimated expected log likelihood decreases. So for a sufficiently large number of Monte Carlo samples, we can ensure that the observed data likelihood increases with high probability.

2.3. Regularization and variable selection

In many cases, it is desirable to model the effects of many features. For instance, Yaari et al. (2013) estimate a 5-mer model with 1024 parameters. Estimating the parameters for a per-target model increases the number of parameters by a factor of four. If the number of sequences in the dataset is small compared to the number of features, the optimization problem in (5) can be ill-posed. For such high-dimensional settings, it is common to use regularization to stabilize our estimates and encourage model structure.

In particular, we may believe that only a small subset of the features affect the mutation rate. Yaari et al. (2013) assume that the nucleotides closest to a position have the most significant effect on its mutation rate: for 5-mer motifs with a small number of observations, they estimate its mutation rate using an offset 3-mer motif. In our method, we use the lasso (Tibshirani, 1996) to encode such prior beliefs and perform variable selection.

To incorporate the lasso, our estimation procedure requires two steps. The first step maximizes the observed log-likelihood with a lasso penalty and thereby performs variable selection. In the second step, we maximize the unpenalized observed log-likelihood with respect to the support of $\boldsymbol{\theta}$ estimated from the first step, constraining all other model parameters to be zero. The goal of the second step is to quantify the uncertainty of our estimates. Fitting an unpenalized model allows us to use traditional inference procedures to obtain "naive" confidence intervals of the model parameters.

In the first step, we split the data into training and validation sets denoted $\mathbf{S}_{\text{obs,train}}$ and $\mathbf{S}_{\text{obs,val}}$, respectively, and maximize the penalized log-likelihood of the training data

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} \ \log \mathcal{L}(\mathbf{S}_{\text{obs,train}}; \boldsymbol{\theta}) - \lambda \|\boldsymbol{\theta}\|_1, \tag{8}$$

where $\lambda > 0$ is a penalty parameter. To solve (8), we use a variant of MCEM: the E-step is the same as before, but we maximize the penalized EM surrogate function during the M-step. The penalized EM surrogate function is simply (7) with a lasso penalty:

$$\frac{1}{E} \sum_{i=1}^{E} \log \mathcal{L}_c \left(\mathbf{S}_{\text{obs,train}}, \boldsymbol{\pi}^{(i)}; \boldsymbol{\theta} \right) - \lambda \|\boldsymbol{\theta}\|_1. \tag{9}$$

This can be maximized using the generalized gradient ascent algorithm given in Algorithm 2 (Nesterov, 2013; Beck and Teboulle, 2009).

Algorithm 2 M-step via generalized gradient ascent

Initialize $\boldsymbol{\theta}$. Choose a step size $\alpha > 0$.
for iteration $k = 1, 2, ...$ until convergence **do**

$$\boldsymbol{\theta} \coloneqq \boldsymbol{\theta} + \alpha \nabla_\theta \frac{1}{E} \sum_{i=1}^{E} \mathcal{L}_c(\mathbf{S}_{\text{obs,train}}, \boldsymbol{\pi}^{(i)}; \boldsymbol{\theta})$$

  for parameter index $j = 1, .., p$ **do**
    $\theta_j \coloneqq \text{sign}(\theta_j) \max(|\theta_j - \lambda|, 0)$
  end for
end for

We tune the penalty parameter λ in (8) by training-validation split. Ideally, we choose the penalty parameter that maximizes the likelihood of the observed validation data. Unfortunately, the likelihood of observed data is computationally intractable. Instead we use the property that, for any $\boldsymbol{\theta}$ and $\boldsymbol{\theta}'$, the difference between the log-likelihoods of the observed data is bounded below by the difference between the expected log-likelihoods of the complete data:

$$\begin{aligned} \log \mathcal{L}(\mathbf{S}_{\text{obs}}; \boldsymbol{\theta}) - \log \mathcal{L}(\mathbf{S}_{\text{obs}}; \boldsymbol{\theta}') \geq \\ \mathbb{E}\left[\log \mathcal{L}_c(\mathbf{S}_{\text{obs}}, \boldsymbol{\pi}; \boldsymbol{\theta}) - \log \mathcal{L}_c(\mathbf{S}_{\text{obs}}, \boldsymbol{\pi}; \boldsymbol{\theta}') \mid \mathbf{S}_{\text{obs}}; \boldsymbol{\theta}'\right]. \end{aligned} \tag{10}$$

The expectation above is taken with respect to the conditional distribution of the mutation orders $\boldsymbol{\pi}$ given the observed data $\mathbf{S}_{\text{obs}}$ and model parameter $\boldsymbol{\theta}'$. Thus the right-hand side can be estimated by sampling mutation orders from the Gibbs sampler in Algorithm 1. If the right-hand side of (10) is positive, then $\boldsymbol{\theta}$ has a higher log-likelihood than $\boldsymbol{\theta}'$ on the validation set. However, if the right-hand side is negative, we do not know how the two parameters compare.

Our proposal for tuning the penalty parameter, Algorithm 3, is based on (10). The algorithm searches across a grid of penalty parameters, from largest to smallest. For consecutive penalty parameters, we estimate the right-hand side of (10) to determine if the smaller penalty parameter has a higher observed log-likelihood. We keep shrinking the penalty parameter until the estimate for the right-hand side of (10) is negative. Since the check based on (10) is conservative, we may end up choosing a penalty parameter that is slightly larger than desired. Nonetheless, our simulation results suggest that this procedure works well in practice.

After we have tuned the penalty parameter and performed variable selection on $\boldsymbol{\theta}$, we refit the model over the entire dataset by maximizing the unpenalized observed log-likelihood (5) over the selected variables. We also estimate confidence intervals for the model parameters, ignoring the fact that we performed model selection during the first step. These confidence intervals are technically invalid since we have peeked at the data twice: once in fitting with a penalized objective and again when refitting with an unpenalized objective. Unfortunately, estimating confidence intervals after model selection is a difficult problem, even in the much simpler case of linear models (Dezeure et al., 2015). Hence some papers use the "naive" approach of fitting a penalized model and then refit-

Algorithm 3 Tuning penalty parameters

Consider a grid of penalty parameters $\lambda_1 > ... > \lambda_K \geq 0$.
Initialize $\lambda_{\text{best}} \coloneqq \lambda_1$. Fit λ_1 to get $\hat{\boldsymbol{\theta}}_{(1)}$.
for iteration $i = 2, ..., K$ **do**
  Solve (8) with $\lambda \equiv \lambda_i$ to get $\hat{\boldsymbol{\theta}}_{(i)}$.
  Estimate via Monte Carlo

$$\eta = \mathbb{E}\left[\log \mathcal{L}_c\left(\mathbf{S}_{\text{obs,val}}, \boldsymbol{\pi}; \hat{\boldsymbol{\theta}}_{(i)}\right) - \log \mathcal{L}_c\left(\mathbf{S}_{\text{obs,val}}, \boldsymbol{\pi}; \hat{\boldsymbol{\theta}}_{(i-1)}\right) \mid \mathbf{S}_{\text{obs,val}}; \hat{\boldsymbol{\theta}}_{(i-1)}\right].$$

  if $\eta > 0$ **then**
    $\lambda_{\text{best}} \coloneqq \lambda_i$
  else
    break
  end if
end for

ting based on the selected variables (Leeb et al., 2015; Hesterberg et al., 2008). Though these confidence intervals are asymptotically valid only under very restrictive conditions, they provide some measure of the uncertainty of our fitted parameters; we investigate the coverage of these intervals via simulation in Section 3. We refer to these intervals as uncertainty intervals in our paper, where 100(1 - α)% uncertainty intervals are constructed using intervals with nominal 100(1 - α)% coverage.

To obtain these uncertainty intervals, we calculate the standard error of our estimates using an estimate of the observed information matrix. Louis (1982) shows that the observed information matrix is related to the complete data likelihood:

$$\begin{aligned} I\left[\boldsymbol{\theta} \mid \mathbf{S}_{\text{obs}}\right] = &-\mathbb{E}\left[\nabla^2_{\boldsymbol{\theta}} \log \mathcal{L}_c(\mathbf{S}_{\text{obs}}, \boldsymbol{\pi}; \boldsymbol{\theta}) \mid \mathbf{S}_{\text{obs}}; \boldsymbol{\theta}\right] \\ &- \mathbb{E}\left[\nabla_{\boldsymbol{\theta}} \log \mathcal{L}_c(\mathbf{S}_{\text{obs}}, \boldsymbol{\pi}; \boldsymbol{\theta}) \left(\nabla_{\boldsymbol{\theta}} \log \mathcal{L}_c(\mathbf{S}_{\text{obs}}, \boldsymbol{\pi}; \boldsymbol{\theta})\right)^\top \mid \mathbf{S}_{\text{obs}}; \boldsymbol{\theta}\right] \\ &+ \mathbb{E}\left[\nabla_{\boldsymbol{\theta}} \log \mathcal{L}_c(\mathbf{S}_{\text{obs}}, \boldsymbol{\pi}; \boldsymbol{\theta}) \mid \mathbf{S}_{\text{obs}}; \boldsymbol{\theta}\right] \mathbb{E}^\top \left[\nabla_{\boldsymbol{\theta}} \log \mathcal{L}_c(\mathbf{S}_{\text{obs}}, \boldsymbol{\pi}; \boldsymbol{\theta}) \mid \mathbf{S}_{\text{obs}}; \boldsymbol{\theta}\right]. \end{aligned}$$

Therefore we can estimate the observed information matrix using samples from the final MCEM iteration and then invert it to obtain uncertainty intervals.

Finally, one caveat of our method is that the two-step procedure is not guaranteed to give estimates of standard errors/uncertainty intervals: The first step of our procedure may choose a penalty parameter such that the estimated information matrix in the second step is not positive definite. This problem occurs primarily in the cases where there is not enough data. We see this behavior in a handful of simulations in Section 3, though we do not observe such behavior in our data analysis. A potential solution is to choose a larger penalty parameter so that the refitting procedure in the second step is performed over a smaller number of model parameters, but we hope to address this issue in a less ad-hoc fashion in future research.

2.4. Examples

By varying the motif dictionary $\mathcal{M}$, our procedure can fit different models of the mutation process. In this section, we list some example models that can be fit using our procedure and discuss the interplay between the motifs included in $\mathcal{M}$ and our feature-selection step. In the simplest case, we can estimate a "k-mer model" (where k is odd) by letting

$$\mathcal{M} = \left\{(m, (k+1)/2) : m \in \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}^k\right\}. \tag{11}$$

The lasso would encourage setting elements in $\boldsymbol{\theta}$ to zero, which means that these k-mer motifs would have the same baseline risk of experiencing a mutation.

In practice, instead of modeling only the effects of k-mers for a fixed k, we may believe that the hazard rate for a position is affected more by positions closer to it. In this case, we model the effect of z-mers of varying length, e.g., $1, 3, ..., k$-mer motifs, with

$$\mathcal{M} = \{(m, (z+1)/2) : m \in \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}^z, z \in 1, 3, \ldots k\}. \tag{12}$$

We refer to this model as "hierarchical", as the elements in $\mathcal{M}$ relate to each other in a nested fashion. By including motifs in a hierarchical fashion, the lasso penalty encourages z-mers with the same inner $(z-2)$-mer to share the same mutation rate. This model formalizes the intuition used by Yaari et al. (2013): they try to estimate the mutation rates of 5-mers but fall back to using a 3-mer sub-motif if that 5-mer does not appear enough times in the data.

As mentioned before, we can add offset motifs to our motif dictionary as previous work suggests the mutation rates depend on upstream or downstream motifs (Rogozin and Kolchanov, 1992; Pham et al., 2003; Yaari et al., 2013). For instance, one can include all the offset motifs that overlap the mutating position in the motif dictionary. We refer to such models as offset k-mer models.

Finally, we can model the hazard rate of motifs mutating to different target nucleotides as in (3). The benefit of having both $\boldsymbol{\theta}$ and $\boldsymbol{\theta}_{\mathtt{N}}$ for $\mathtt{N} \in \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$ in the model is that the lasso penalty will encourage hazard rates for the different target nucleotides to be the same if they share the same motif.

Many of these example models are overparameterized in order to obtain some desired sparsity pattern. Such overparameterized models may have singular information matrices during the refitting procedure, which means that we cannot get uncertainty intervals for the model parameters $\boldsymbol{\theta}$. However we can still get confidence intervals for the parameters $\boldsymbol{\theta}_{\text{agg}} = \boldsymbol{A}\boldsymbol{\theta}$ associated with the simple k-mer model, where $\boldsymbol{A}$ is a matrix that aggregates hierarchical motifs into a single k-mer. Since this aggregate k-mer model is identifiable, we can get uncertainty intervals for $\boldsymbol{\theta}_{\text{agg}}$: we calculate the pseudo-inverse $\boldsymbol{I}^-$ of the (estimated) information matrix and then use $\boldsymbol{A}\boldsymbol{I}^-\boldsymbol{A}^\top$ to get an estimate of the covariance matrix of $\boldsymbol{\theta}_{\text{agg}}$.

3. Simulation Results

We now present a simulation study of our procedure, including a comparison to the current state-of-the-art method `SHazaM` (Yaari et al., 2013).

3.1. Understanding the effect of various models and settings

We fit the following three models to simulated data:

- 3-mer model: the hazard rate modeled by (2) with motif dictionary (11) where $k = 3$,
- 3-mer per-target model: the hazard rate modeled by (3) with motif dictionary (11) where $k = 3$,
- 2,3-mer model: the hazard rate modeled by (2) with motif dictionary

$$\mathcal{M} = \left\{(m, j') : m \in \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}^2, j' \in \{1, 2\}\right\} \cup \left\{(m, 2) : m \in \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}^3\right\}.$$

To understand how dataset composition affects the performance of our procedure, we simulate different datasets by varying the sample sizes, sparsity levels, and effect sizes.

For each 3-mer model we fit, we generate the true $\boldsymbol{\theta}$ according to same hierarchical structure. Let the model parameters corresponding to the motif m be θ_m and corresponding to motif m with target nucleotide `N` be $\theta_{m\to\mathtt{N}}$. To generate the true $\boldsymbol{\theta}$ parameters, we randomly draw values from the mouse somatic hypermutation targeting model `MK_RS5NF` of Cui et al. (2016); we refer to these parameters as $\boldsymbol{\theta}_{\mathtt{MK}}$. To obtain the desired sparsity level, we randomly select a portion of the parameters to zero out; for per-target parameters, we set $\theta_{m\to\mathtt{N}}$ to $\log 1/3$ for all possible values of `N`. We scale the model parameters appropriately to control the effect size.

To obtain synthetic data that reflects the experimental data in Cui et al. (2016) analyzed in Section 4, we use the following setup. We first generate a set of κ-light chain mouse immunoglobulin genes using `partis` (Ralph and Matsen IV, 2016a,b) by drawing a set of genes from the IMGT database (Lefranc et al., 1999) and simulating an observation frequency for each. The average length of the naive sequences is around 290 nucleotides. We use the survival model to mutate between 1% and 5% of the positions of each naive sequence, obtaining a collection of simulated BCR sequences. Conditional on their naive sequences, BCR sequences mutate independently.

We vary sparsity, effect size, and sample size as follows. We generate the true $\boldsymbol{\theta}$ parameters with 25%, 50%, and 100% non-zero elements. We also consider different effect sizes by scaling $\boldsymbol{\theta}$ such that its variance is 50%, 100%, and 200% of the variance of the values in $\boldsymbol{\theta}_{\mathtt{MK}}$. Finally, we fit the model using 100, 200, and 400 mutated BCR sequences. When varying one of the simulation settings, we fix the other simulation settings to the middle value (e.g., we vary number of samples but keep the effect size at 100% and the number of non-zero elements at 50%). We replicate each simulation ten times.

To determine the optimal penalty parameter for `samm`, we randomly select 20% of the gene subgroups (Lefranc, 2014) and reserve their corresponding B-cell receptors to make a validation set. We then apply Algorithm 3 over a decreasing sequence of penalty parameter-values: $10^{-1}, 10^{-1.5}, 10^{-2}$.... For each penalty parameter-value, we run a maximum of 10 MCEM iterations. Mutation orders are sampled from each Gibbs sampler run every eight sweeps, after an initial burn-in period of 16 Gibbs sweeps. For each E-step, we sample four mutation orders and continue to double the number of sampled mutation orders if the proposed estimate is not accepted by ascent-based MCEM.

We assess the performance of our procedure using three measures. These performance metrics are all calculated with respect to the aggregate model since our fitting procedure is overparameterized. We calculate the relative $\boldsymbol{\theta}$ error, defined as $\|\boldsymbol{\theta} - \boldsymbol{\theta}^*\|_2 / \|\boldsymbol{\theta}^*\|_2$, to see how close the estimated parameters are to the truth. We also calculate Kendall's tau coefficient to see how well our procedure ranks the motifs in terms of their mutabilities. Finally we calculate the coverage of our approximate 95% uncertainty intervals. We define the average coverage as the proportion of aggregate model parameters where the uncertainty intervals covered the true value. The coverage calculations only involve aggregate parameters not zeroed out by our models.

These simulations demonstrate that our estimation procedure performs as expected (Figure 3). As the sample size and effect size increase, the relative $\boldsymbol{\theta}$ error decreases and the rank correlation increases. On the other hand, as the percent of non-zero elements increases, both the relative $\boldsymbol{\theta}$ error and rank correlation increase. The error increases because there are more parameters to estimate. The increase in rank correlation is likely an artifact of how the metric is calculated, as Kendall's tau removes ties from the calculations. In particular, as the percent of non-zero elements increases, the number of ties in the data decreases, so the rank correlation seems to increase. In all the plots, we see that the 3-mer per-target model tends to be the most difficult to estimate. This is expected as it contains 256 parameters whereas the 3-mer model only has 64 parameters.

Our simulations show that the coverages for the 3-mer and the 2,3-mer models are close to 95%, which is surprising as our uncertainty intervals ignore the double-peeking issue (Figure 3). Zhao, Shojaie and Witten (2017) explain why this procedure might work: under certain assumptions, the variables selected by the lasso are deterministic with high probability, so using the lasso to select variables does not really constitute as peeking at the data twice.

However, the coverage of the 3-mer per-target is much lower, dropping below 70% in certain settings (Figure 3). We suspect that the low coverage is mainly due to a lack of data, as the coverage improves with the number of samples. When there is a small number of samples compared to the number of parameters, our method may only provide a reasonable ranking of how mutable the motifs are but may not provide good estimates and uncertainty intervals. In fact, without enough data, our procedure may not be able to estimate uncertainty intervals at all. For the 3-mer per-target model, three simulation replicates fail to obtain confidence intervals, indicated by asterisks in Figure 3.

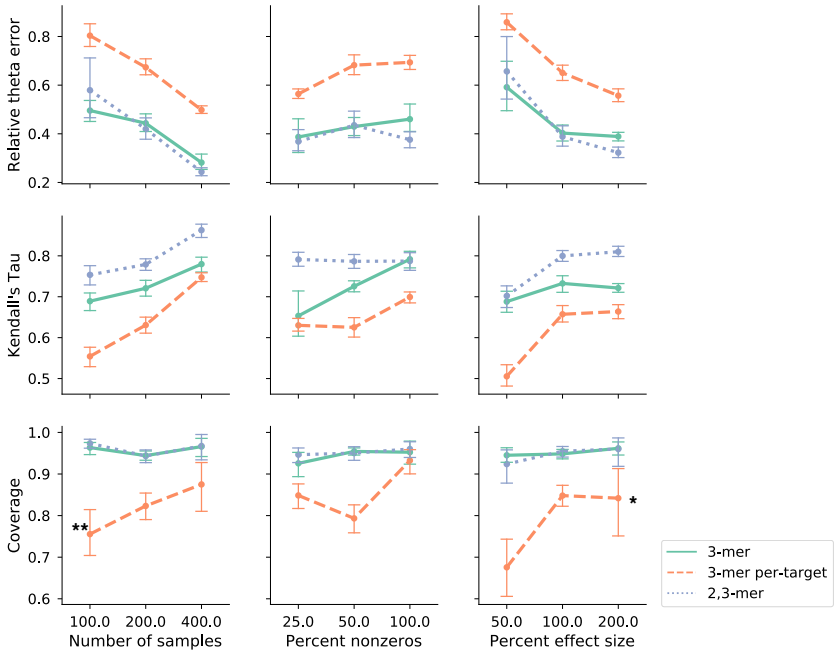


FIG 3. *Relative error, correlation and coverage under different simulations settings for 3mer, 3mer per target and 2,3mer models. Each asterisk indicates a single simulation replicate (of ten for each model and setting) where the estimated information matrix is not positive definite, e.g. the uncertainty intervals cannot be established.*

3.2. Comparing to SHazaM

In this section, we compare the performance of `samm` to `SHazaM` on simulated data. Since `SHazaM` only estimates the effect of 5-mer motifs, we simulate data such that the mutation rate at a specific site depends on the 5-mer centered at that position and the target nucleotide. We simulate BCR sequences from 4 or 8 mice. For each mouse, we generate a separate set of mouse immunoglobulin genes using the same procedure as in Section 3.1. From these naive sequences, we simulate the mutation process independently to generate BCR sequences. We use two methods to simulate the mutation process:

- **Survival Simulation**: We generate model parameters $\boldsymbol{\theta}$ by resampling the values from $\boldsymbol{\theta}_{\texttt{MK}}$ into a 3,5-mer per-target model structure. We then mutate the naive sequences according to the survival model.
- `SHMulate` **Simulation**: We use $\boldsymbol{\theta}_{\texttt{MK}}$ and mutate the naive sequences using the `SHMulate` function in the `SHazaM` package (Yaari et al., 2013; Gupta et al., 2015). `SHMulate` simulates the mutation process using a procedure that is similar to a survival model. However the exact calculations differ somewhat (e.g. it does not allow the mutation process to create stop codons).

The simulations are run until 1–5% of the sequence is mutated. This mutation rate is on the low end for affinity-matured BCR sequences (compare the 3× higher rate in He et al., 2014), giving `SHazaM` a slight edge since many fewer motif-modifying mutations (not accounted for in that package) will occur in simulation.

We fit a 3,5-mer per-target `samm` model using the same procedure as in Section 3.1. We compare our model fits to `SHazaM` by averaging the result over ten simulation replicates. `SHazaM` should have another advantage in the `SHMulate` simulations since the $\boldsymbol{\theta}_{\texttt{MK}}$ was estimated using `SHazaM` on a separate BCR dataset and `SHazaM` uses some prior beliefs about the model structure. In particular, `SHazaM` assumes that 5-mer motifs that share certain upstream/downstream nucleotides have similar mutabilities.

We measure model performance by the relative $\boldsymbol{\theta}$ error and rank correlation as before. When simulating 2000 BCRs from 4 mice using the survival model, `samm` significantly outperforms `SHazaM` (Table 1). The model fits are presented in more detail in Figure 4 (top). For negative $\boldsymbol{\theta}$ values, both `samm` and `SHazaM` are biased towards zero, though `SHazaM` tends to be more so. For positive $\boldsymbol{\theta}$ values, `samm` is nearly unbiased while `SHazaM` is still biased towards zero. The two methods probably have trouble estimating negative values since we only observe a small number of mutations per sequence. Thus the data is more informative for finding motifs with high mutation rates rather than those with low mutation rates.

When we simulate 2000 BCRs from 4 mice using `SHMulate`, the two estimation methods are comparable. Given that `SHMulate` and the survival model use similar procedures to simulate mutabilities, we hypothesize that `SHazaM` does well in this scenario because $\boldsymbol{\theta}_{\texttt{MK}}$ was originally estimated using `SHazaM`. We therefore also simulate a larger dataset with 4000 BCRs from 8 mice using

TABLE 1
Comparison of `samm` *and* `SHazaM` *on simulated B-cell receptor data. Standard errors are given in parentheses.*

	Simulator	Model	Relative $\boldsymbol{\theta}$ error	Kendall's tau
4 mice, 2000 BCRs	survival model	`samm`	0.591 (0.005)	0.624 (0.003)
		`SHazaM`	0.726 (0.006)	0.511 (0.004)
	`SHMulate`*	`samm`	0.490 (0.005)	0.681 (0.003)
		`SHazaM`	0.492 (0.005)	0.686 (0.003)
8 mice, 4000 BCRs	`SHMulate`	`samm`	0.413 (0.003)	0.737 (0.002)
		`SHazaM`	0.429 (0.002)	0.734 (0.001)

* In one of the ten simulation replicates, `SHazaM` failed to estimate the mutability of some motifs since they do not appear often enough in the dataset. We omit this simulation replicate.

`SHMulate`. In this case, `samm` significantly outperforms `SHazaM` in terms of relative $\boldsymbol{\theta}$ error. This suggests that `SHazaM` only does better than `samm` in the 2000 BCR scenario by relying on known model structure.

4. Data analysis

We fit models to the BCR sequence data obtained from a vaccination study of four transgenic mice published in Cui et al. (2016). In this experimental setting, the substitutions present in the κ-light chain sequences are unlikely to be affected by natural selection on BCR function. Thus we restrict our analysis to only κ-light chain data in order to estimate somatic hypermutation rates, rather than a combination of somatic hypermutation and selection (Yaari, Uduman and Kleinstein, 2012; McCoy et al., 2015; Yaari et al., 2015). A single naive sequence can give rise to many different B-cell receptors by somatic hypermutation, forming a so-called "clonal family" which may have varying level of shared evolutionary history. We use `partis` (Ralph and Matsen IV, 2016a) to assign mutated sequences to clonal families and infer the most likely naive sequence in each family. The resulting data has the composition shown in Table 2. To mitigate double-counting mutations, we sample a single mutated sequence from each clonal family. In Section C.1 in the Appendix, we show that this gives more accurate estimates of the unknown $\boldsymbol{\theta}$ than other approaches.

TABLE 2
Statistics of processed κ-light chain data from Cui et al. (2016).

Number of mutated sequences	16,971
Number of clonal families	2,593
Median mutated sequence length	282
Average mutation frequency (%)	2.58
Number of 5-mers in naive sequences	1,014

We fit a 3,5-mer model using `samm` using the same settings as before (Figure 5). The $\boldsymbol{\theta}$ estimate has a block-like and 4-fold-repetitive pattern because

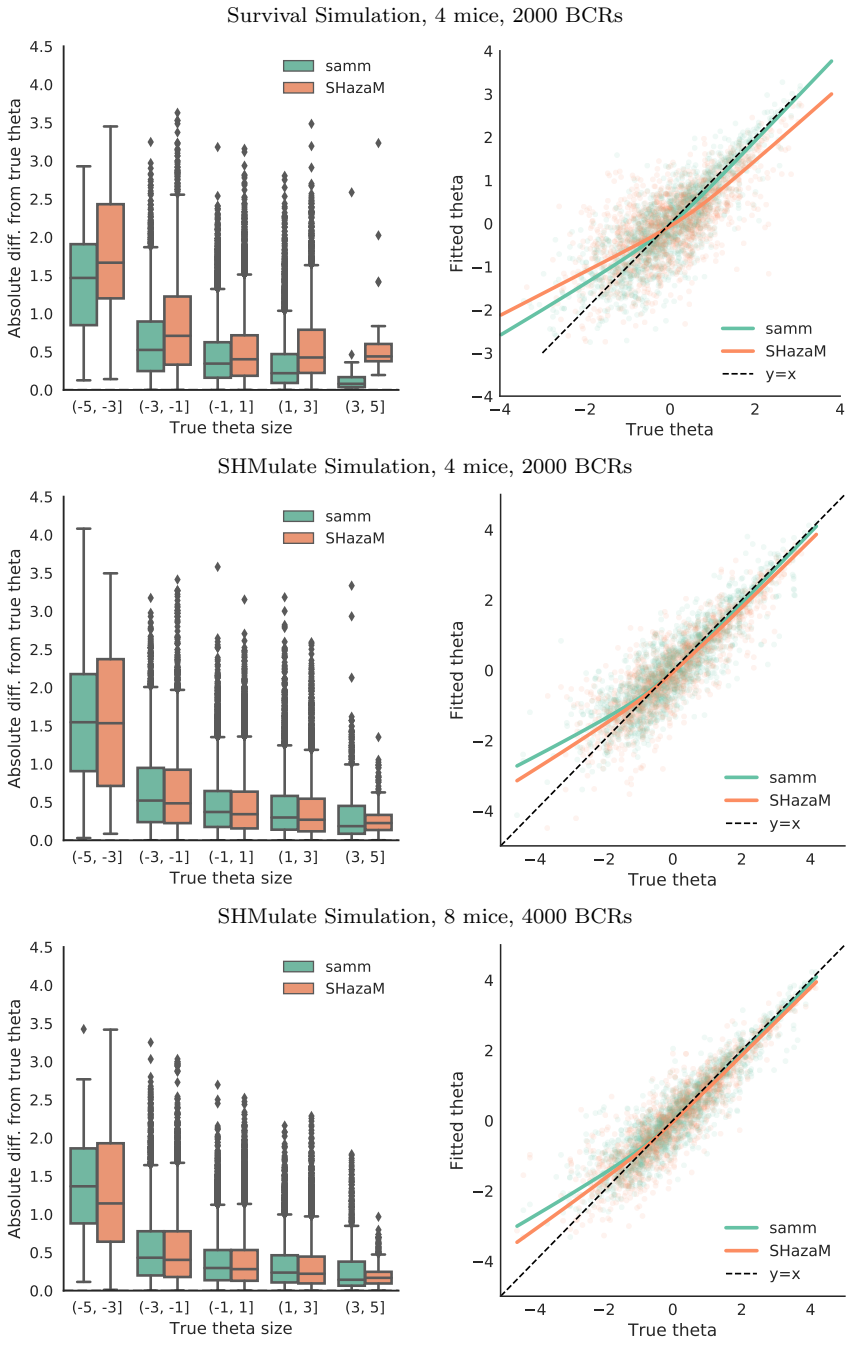


FIG 4. *Left: boxplots of the absolute differences between fitted and true* $\boldsymbol{\theta}$ *values. Right: scatter plots and LOWESS fits of fitted vs. true* $\boldsymbol{\theta}$ *values. All values for* $\boldsymbol{\theta}$ *are median-centered.*

many 5-mer motifs were zeroed out during the lasso step. The 95% uncertainty intervals suggest that many motifs have a marked nonzero effect.

Our model recovered many of the well-known "hot" (more mutable) and "cold" spots (less mutable k-mers), which are denoted by the red, blue, and green bars in Figure 5. Hot/cold motifs are typically denoted with an underline indicating which position is mutating and represented by degenerate bases $\texttt{W} \in \{\texttt{A}, \texttt{T}\}$, $\texttt{R} \in \{\texttt{A}, \texttt{G}\}$, $\texttt{Y} \in \{\texttt{C}, \texttt{T}\}$, $\texttt{S} \in \{\texttt{C}, \texttt{G}\}$. We confirm that many highly mutable 5-mer motifs match the classical hot spot motif WR$\underline{\texttt{C}}$ and its reverse complement $\underline{\texttt{G}}$YW (since the mutation process can happen on either DNA strand) (Rogozin and Diaz, 2004). We also confirmed that many less mutable 5-mer motifs match the canonical cold spot SY$\underline{\texttt{C}}$/$\underline{\texttt{G}}$RS (Yaari et al., 2013).

However our model reveals shortcomings with the current hot and cold spot definitions. Our estimates show that there is significant variability in the mutabilities of motifs, even if they contain the same hot or cold spot motif. For instance, in the established literature the AC$\underline{\texttt{G}}$GG motif is considered to be a cold spot since it is of the form $\underline{\texttt{G}}$RS. We estimate its θ value to be very large relative to the other θ values, suggesting that it is actually a hot spot. In addition, the classic hot spots with a central T nucleotide actually had very low mutability estimates; this suggests that using the well-known W$\underline{\texttt{A}}$/$\underline{\texttt{T}}$W to identify hot spots may not be appropriate.

Finally, our model suggests that `samm` can be used to discover new hot and cold spots. In particular, the mutability of the A$\underline{\texttt{C}}$G motif is similar to the WR$\underline{\texttt{C}}$ hot spot, indicating perhaps the immediate upstream R has the highest effect on mutability at a C site. A well-defined inferential procedure to determine significant collections of hot and cold spots with ample support from the data will require additional future work.

For comparison, we also fit `SHazaM` on the same data. Both `SHazaM` and `samm` use the data to determine the degrees of freedom to use in fitting $\boldsymbol{\theta}$, resulting in the number of unique $\boldsymbol{\theta}$ values fit to be less than the saturated model size of 1024 for a 5-mer model. `SHazaM` estimated 1015 unique θ values out of a maximum of 1024 while `samm` only estimated 382 unique θ values. Visually, their estimates look similar to `samm`, though they tend to be more "spiky" (Figure 6). So in terms of model interpretability, `samm` seems to be preferable as it produces much more parsimonious models.

Ideally, we would be able to compare the two models in terms of their observed data likelihood on a test set. However calculating the observed data likelihood is computationally difficult. Instead, we calculated the EM surrogate function in (10) as a proxy: if the difference of the EM surrogate functions is positive, then `samm` performs better than `SHazaM`, and if the value is negative, the test is inconclusive. We also tried using the $\boldsymbol{\theta}$ estimate from `samm` as a reference in (10) to see if `SHazaM` performed better than `samm`. Unfortunately, our tests based on the EM surrogate function were inconclusive – the difference of the EM surrogate functions were both negative. We hope to come up with better ways to compare models in the future.

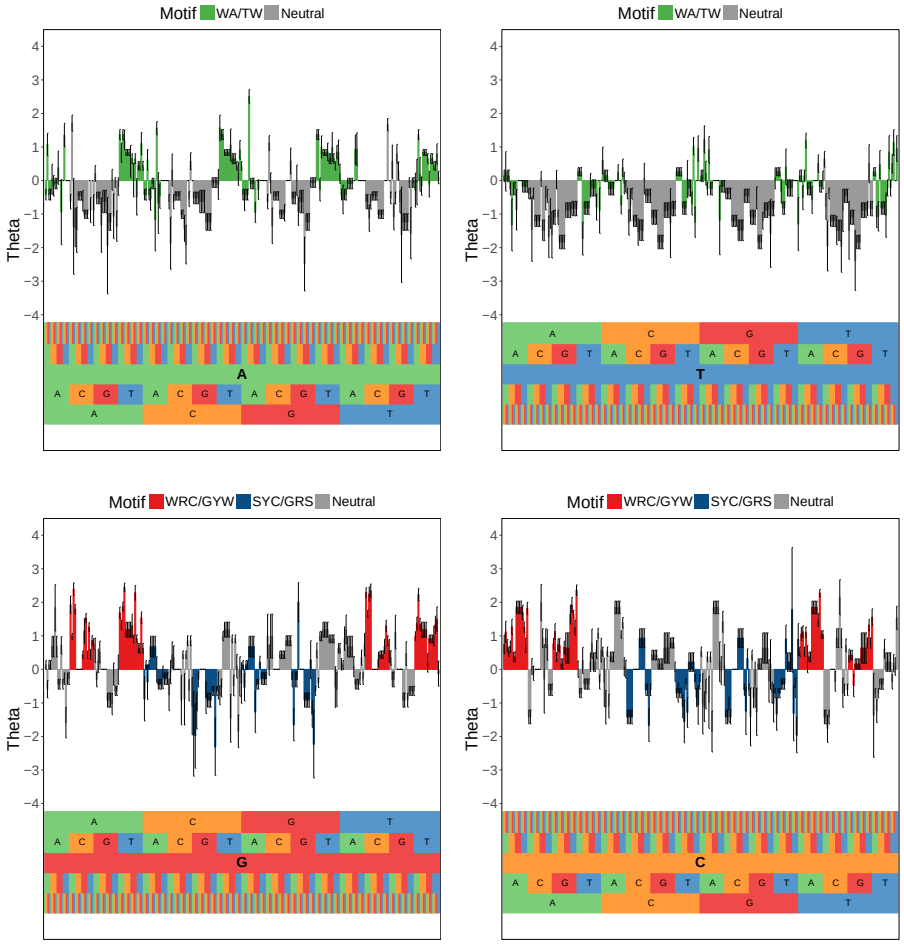
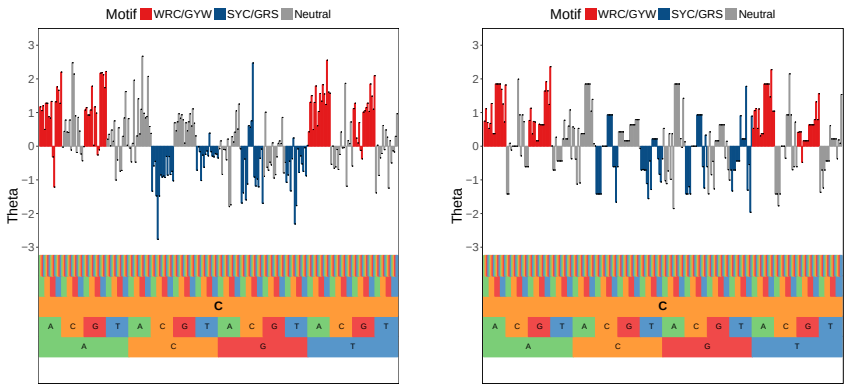


FIG 5. *Somatic hypermutation model for mouse light chains estimated by* `samm` *for a 3,5-mer model. Plots depict the estimated* θ *of 5-mer motifs centered on the bases* `A` *(top left),* `T` *(top right),* `G` *(bottom left), and* `C` *(bottom right). A negative theta value means a suppression of mutation relative to the baseline hazard, whereas a positive means an enhancement. The motif corresponding to an x-axis position can be read from bottom to top. Well-known hot spots,* `WRC`/`GYW` *and* `WA`/`TW`*, are colored red and green, respectively. The well-known cold spot* `SYC`/`GRS` *is colored blue. All other motifs are colored grey. The 95% uncertainty intervals for the* θ *estimates are depicted by black lines in the center of each bar.*



(a) `SHazaM`: 1015 unique θ values (b) `samm`: 382 unique θ values

FIG 6. *A side-by-side comparison of fitted θ values from `SHazaM` (left) and `samm` (right). `samm` tends to fit more parsimonious models compared to `SHazaM`, so the left plot looks more "spiky" than the right one.*

5. Discussion

We have modeled somatic hypermutation of BCR sequences using Cox proportional hazards. Due to the context-dependence of mutation rates, we must take into account the unknown mutation order to compute the full likelihood. To deal with this missing data, we used MCEM, where we marginalize over the possible mutation orders using Markov chain Monte Carlo. Unlike current methods, our regression framework can model the effect of arbitrary features, such as varying motif lengths and sequence positions. In this paper, we use the lasso to perform feature selection and stabilize our estimates in high-dimensional settings. One can easily extend this approach to use other sparsity-inducing penalties to reflect other prior beliefs about the model structure. We show that `samm` achieves better performance than the state-of-the-art method under a variety of simulation settings.

There are a few limitations with our current method. We currently subsample our data significantly to ensure our training set is composed of independent observations. This would not be necessary if we were able to perform accurate phylogenetic ancestral sequence estimation using context-sensitive models. In addition, our method returns "uncertainty" intervals rather than confidence intervals since there are no guarantees on their nominal coverage. Simulations show that our uncertainty intervals are close to their nominal coverage levels if there is a sufficient amount of data, but better methods may be available.

While the present analysis only considers sequence context, other biologically-motivated features may be just as informative: nucleotide position, proximity to other contexts, etc. By incorporating other types of features into the model, we may be able to help verify or find problems with the currently accepted model of somatic hypermutation (Methot and Di Noia, 2017).

Finally, our model can be used in other contexts to model other biological processes. For instance, our method could be used to model the rate of single-nucleotide polymorphisms (Aggarwala and Voight, 2016) and transcription-factor binding (Zhou and Liu, 2004).

Acknowledgments

Jean Feng was supported by NIH grants DP5OD019820 and T32CA206089. Noah Simon was supported by NIH grant DP5OD019820. David Shaw, Vladimir Minin, and Frederick Matsen were supported by NIH grants U19-AI117891 and R01-GM113246; David Shaw was also supported by R01-AI12096. The research of Frederick Matsen was supported in part by a Faculty Scholar grant from the Howard Hughes Medical Institute and the Simons Foundation.

We are grateful to Duncan Ralph for assistance performing sequence annotation, clustering and simulating germline repertoires. We would like to thank the Kleinstein lab for generously sharing DNA sequences, and especially to Jason Vander Heiden for providing us with preprocessed versions of their sequence data.

References

Aggarwala, V. and Voight, B. F. (2016). An expanded sequence context model broadly explains variability in polymorphism levels across the human genome. *Nature Genetics* **48** 349–355.

Beck, A. and Teboulle, M. (2009). A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences* **2** 183–202.

Bonissone, S. R. and Pevzner, P. A. (2015). Immunoglobulin classification using the colored antibody graph. In *Research in Computational Molecular Biology* (T. M. Przytycka, ed.). *Lecture Notes in Computer Science* 44–59. Springer International Publishing.

Caffo, B. S., Jank, W. and Jones, G. L. (2005). Ascent-based Monte Carlo expectation–maximization. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **67** 235–251.

Chahwan, R., Edelmann, W., Scharff, M. D. and Roa, S. (2012). AIDing antibody diversity by error-prone mismatch repair. *Seminars in Immunology* **24** 293–300.

Cohen, R. M., Kleinstein, S. H. and Louzoun, Y. (2011). Somatic hypermutation targeting is influenced by location within the immunoglobulin V region. *Molecular Immunology* **48** 1477–1483.

Cowell, L. G. and Kepler, T. B. (2000). The nucleotide-replacement spectrum under somatic hypermutation exhibits microsequence dependence that is strand-symmetric and distinct from that under germline mutation. *The Journal of Immunology* **164** 1971–1976.

Cui, A., Di Niro, R., Vander Heiden, J. A., Briggs, A. W., Adams, K., Gilbert, T., O'Connor, K. C., Vigneault, F., Shlomchik, M. J. and Kleinstein, S. H. (2016). A model of somatic hypermutation targeting in mice based on high-throughput Ig sequencing data. *The Journal of Immunology* **197** 3566–3574.

Dempster, A. P., Laird, N. M. and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 1–38.

Dezeure, R., Bühlmann, P., Meier, L., Meinshausen, N. et al. (2015). High-dimensional inference: confidence intervals, p-values and R-software hdi. *Statistical science* **30** 533–558.

Dunn-Walters, D. K., Dogan, A., Boursier, L., MacDonald, C. M. and Spencer, J. (1998). Base-specific sequences that bias somatic hypermutation deduced by analysis of out-of-frame human IgVH genes. *The Journal of Immunology* **160** 2360–2364.

Elhanati, Y., Sethna, Z., Marcou, Q., Callan, C. G. Jr, Mora, T. and Walczak, A. M. (2015). Inferring processes underlying B-cell repertoire diversity. *Philosophical Transactions of the Royal Society B: Biological Sciences* **370**.

Farris, J. S. (1970). Methods for computing Wagner trees. *Systematic Zoology* **19** 83–92.

Felsenstein, J. (2003). *Inferring phylogenies*. Sinauer Associates, Sunderland, MA.

Felsenstein, J. (2005). PHYLIP (Phylogeny Inference Package) version 3.6. Distributed by the author. Department of Genome Sciences, University of Washington, Seattle.

Goggins, W. B., Finkelstein, D. M., Schoenfeld, D. A. and Zaslavsky, A. M. (1998). A Markov chain Monte Carlo EM algorithm for analyzing interval-censored data under the Cox proportional hazards model. *Biometrics* 1498–1507.

Gupta, N. T., Vander Heiden, J. A., Uduman, M., Gadala-Maria, D., Yaari, G. and Kleinstein, S. H. (2015). Change-O: a toolkit for analyzing large-scale B cell immunoglobulin repertoire sequencing data. *Bioinformatics* **31** 3356–3358.

Haynes, B. F., Kelsoe, G., Harrison, S. C. and Kepler, T. B. (2012). B-cell-lineage immunogen design in vaccine development with HIV-1 as a case study. *Nature Biotechnology* **30** 423–433.

He, L., Sok, D., Azadnia, P., Hsueh, J., Landais, E., Simek, M., Koff, W. C., Poignard, P., Burton, D. R. and Zhu, J. (2014). Toward a more accurate view of human B-cell repertoire by next-generation sequencing, unbiased repertoire capture and single-molecule barcoding. *Scientific Reports* **4** 6778.

Hershberg, U., Uduman, M., Shlomchik, M. J. and Kleinstein, S. H. (2008). Improved methods for detecting selection by mutation analysis of Ig V region sequences. *International Immunology* **20** 683–694.

Hesterberg, T., Choi, N. H., Meier, L., Fraley, C. et al. (2008). Least

angle and ℓ_1 penalized regression: A review. *Statistics Surveys* **2** 61–93.

Ho, S. Y. W. and Jermiin, L. S. (2004). Tracing the decay of the historical signal in biological sequence data. *Systematic Biology* **53** 628–637.

Hobolth, A. (2008). A Markov chain Monte Carlo expectation maximization algorithm for statistical analysis of DNA sequence evolution with neighbor-dependent substitution rates. *Journal of Computational and Graphical Statistics* **17** 138–162.

Hoehn, K. B., Lunter, G. and Pybus, O. G. (2017). A phylogenetic codon substitution model for antibody lineages. *Genetics* **206** 417–427.

Hwang, D. G. and Green, P. (2004). Bayesian Markov chain Monte Carlo sequence analysis reveals varying neutral substitution patterns in mammalian evolution. *Proceedings of the National Academy of Sciences USA* **101** 13994–14001.

Hwang, J. K., Wang, C., Du, Z., Meyers, R. M., Kepler, T. B., Neuberg, D., Kwong, P. D., Mascola, J. R., Joyce, M. G., Bonsignori, M., Haynes, B. F., Yeap, L.-S. and Alt, F. W. (2017). Sequence intrinsic somatic mutation mechanisms contribute to affinity maturation of VRC01-class HIV-1 broadly neutralizing antibodies. *Proceedings of the National Academy of Sciences USA*.

Kalbfleisch, J. D. and Prentice, R. L. (2011). *The statistical analysis of failure time data* **360**. John Wiley & Sons.

Leeb, H., Pötscher, B. M., Ewald, K. et al. (2015). On various confidence intervals post-model-selection. *Statistical Science* **30** 216–227.

Lefranc, M.-P. (2014). Immunoglobulins: 25 years of immunoinformatics and IMGT-ONTOLOGY. *Biomolecules* **4** 1102–1139.

Lefranc, M.-P., Giudicelli, V., Ginestoux, C., Bodmer, J., Müller, W., Bontrop, R., Lemaitre, M., Malik, A., Barbié, V. and Chaume, D. (1999). IMGT, the international ImMunoGeneTics database. *Nucleic Acids Research* **27** 209–212.

Louis, T. A. (1982). Finding the observed information matrix when using the EM algorithm. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 226–233.

McCoy, C. O., Bedford, T., Minin, V. N., Bradley, P., Robins, H. and Matsen, F. A. IV (2015). Quantifying evolutionary constraints on B-cell affinity maturation. *Philosophical Transactions of the Royal Society B: Biological Sciences* **370**.

Methot, S. P. and Di Noia, J. M. (2017). Chapter two - molecular mechanisms of somatic hypermutation and class switch recombination. In *Advances in Immunology*, (Frederick W. Alt, ed.) **133** 37–87. Academic Press.

Nesterov, Y. (2013). Gradient methods for minimizing composite objective functions. *Mathematical Programming* **140** 125–161.

Pham, P., Bransteitter, R., Petruska, J. and Goodman, M. F. (2003). Processive AID-catalysed cytosine deamination on single-stranded DNA simulates somatic hypermutation. *Nature* **424** 103–107.

Ralph, D. K. and Matsen IV, F. A. (2016a). Consistency of VDJ rearrangement and substitution parameters enables accurate B cell receptor sequence

annotation. *PLOS Computational Biology* **12** 1–25.

Ralph, D. K. and Matsen IV, F. A. (2016b). Likelihood-based inference of B cell clonal families. *PLOS Computational Biology* **12** e1005086.

Rogozin, I. B. and Diaz, M. (2004). Cutting edge: DGYW/WRCH is a better predictor of mutability at G: C bases in Ig hypermutation than the widely accepted RGYW/WRCY motif and probably reflects a two-step Activation-Induced Cytidine Deaminase-triggered process. *The Journal of Immunology*.

Rogozin, I. B. and Kolchanov, N. A. (1992). Somatic hypermutagenesis in immunoglobulin genes. II. Influence of neighbouring base sequences on mutagenesis. *Biochimica et Biophysica Acta* **1171** 11–18.

Rogozin, I. B., Pavlov, Y. I., Bebenek, K., Matsuda, T. and Kunkel, T. A. (2001). Somatic mutation hotspots correlate with DNA polymerase η error spectrum. *Nature Immunology* **2** 530–536.

Schatz, D. G. and Ji, Y. (2011). Recombination centres and the orchestration of V (D) J recombination. *Nature Reviews Immunology* **11** 251–263.

Sheng, Z., Schramm, C. A., Kong, R., NISC Comparative Sequencing Program, Mullikin, J. C., Mascola, J. R., Kwong, P. D. and Shapiro, L. (2017). Gene-specific substitution profiles describe the types and frequencies of amino acid changes during antibody somatic hypermutation. *Frontiers in Immunology* **8** 537.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 267–288.

Tibshirani, R. et al. (1997). The lasso method for variable selection in the Cox model. *Statistics in Medicine* **16** 385–395.

Tonegawa, S. (1983). Somatic generation of antibody diversity. *Nature* **302** 575–581.

Uduman, M., Yaari, G., Hershberg, U., Stern, J. A., Shlomchik, M. J. and Kleinstein, S. H. (2011). Detecting selection in immunoglobulin sequences. *Nucleic Acids Research* **39** W499–504.

Wei, G. C. and Tanner, M. A. (1990). A Monte Carlo implementation of the EM algorithm and the poor man's data augmentation algorithms. *Journal of the American Statistical Association* **85** 699–704.

Yaari, G. and Kleinstein, S. H. (2015). Practical guidelines for B-cell receptor repertoire sequencing analysis. *Genome Medicine* **7** 121.

Yaari, G., Uduman, M. and Kleinstein, S. H. (2012). Quantifying selection in high-throughput Immunoglobulin sequencing data sets. *Nucleic Acids Research* **40** e134.

Yaari, G., Vander Heiden, J. A., Uduman, M., Gadala-Maria, D., Gupta, N., Stern, J. N. H., O'Connor, K. C., Hafler, D. A., Laserson, U., Vigneault, F. and Kleinstein, S. H. (2013). Models of somatic hypermutation targeting and substitution based on synonymous mutations from high-throughput immunoglobulin sequencing data. *Frontiers in Immunology* **4** 358.

Yaari, G., Benichou, J. I. C., Vander Heiden, J. A., Kleinstein, S. H. and Louzoun, Y. (2015). The mutation patterns in B-cell immunoglobulin receptors reflect the influence of selection acting at multiple time-scales.

Philosophical Transactions of the Royal Society B: Biological Sciences **370**.

Yeap, L.-S., Hwang, J. K., Du, Z., Meyers, R. M., Meng, F.-L., Jakubauskaitė, A., Liu, M., Mani, V., Neuberg, D., Kepler, T. B., Wang, J. H. and Alt, F. W. (2015). Sequence-intrinsic mechanisms that target AID mutational outcomes on antibody genes. *Cell*.

Zhao, S., Shojaie, A. and Witten, D. (2017). In defense of the indefensible: a very naive approach to high-dimensional inference. *arXiv preprint arXiv:1705.05543*.

Zhou, Q. and Liu, J. S. (2004). Modeling within-motif dependence for transcription factor binding site predictions. *Bioinformatics* **20** 909–916.

Appendix A: Proof of marginal likelihood

We now prove the statement in Section 2 that the marginal likelihood of $\boldsymbol{\theta}$ is given by (4) and only depends on the mutation order $\boldsymbol{\pi}_{1:n}$.

Proof. Suppose the mutation times are observed, so u_i is the time of the i-th mutation. Then the conditional probability of observing a mutation order $\boldsymbol{\pi}_{1:n}$ given mutation times $\boldsymbol{u} = (u_1, ..., u_n)$ can be written as

$$\Pr\left(\boldsymbol{\pi}_{1:n}|\boldsymbol{u};\boldsymbol{\theta},h_0\right) = \prod_{i=1}^{n} \Pr\left(\pi_i|\boldsymbol{\pi}_{1:i-1},u_{i-1},u_i;\boldsymbol{\theta},h_0\right) \tag{13}$$

$$= \prod_{i=1}^{n} \frac{\Pr\left(\pi_i,u_i|\boldsymbol{\pi}_{1:i-1},u_{i-1};\boldsymbol{\theta},h_0\right)}{\sum_{q\in R(\boldsymbol{\pi}_{1:i-1})}\Pr\left(q,u_i|\boldsymbol{\pi}_{1:i-1},u_{i-1};\boldsymbol{\theta},h_0\right),} \tag{14}$$

where the conditional probability of observing a mutation in position q at time u_i is defined as

$$\Pr\left(q,u_i|\boldsymbol{\pi}_{1:i-1},u_{i-1};\boldsymbol{\theta},h_0\right) \tag{15}$$

$$= h_0(u_i)\exp\left(\boldsymbol{\theta}^\top\psi_q(S(\boldsymbol{\pi}_{1:i-1}))\right)\times \tag{16}$$

$$\exp\left(-\sum_{q'\in R(\boldsymbol{\pi}_{1:i-1})}\exp\left(\boldsymbol{\theta}^\top\psi_{q'}(S(\boldsymbol{\pi}_{1:i-1}))\right)\int_{u_{i-1}}^{u_i}h_0(t)dt\right). \tag{17}$$

Notice that in (15), the terms $h_0(u_i)$ and (17) do not depend on q. So plugging (15) into (14), these two terms cancel and we get

$$\Pr\left(\boldsymbol{\pi}_{1:n}|\boldsymbol{u};\boldsymbol{\theta},h_0\right) = \prod_{i=1}^{n}\frac{\exp(\boldsymbol{\theta}^\top\psi_{\pi_i}(S(\boldsymbol{\pi}_{1:i-1})))}{\sum_{q\in R(\boldsymbol{\pi}_{1:i-1})}\exp(\boldsymbol{\theta}^\top\psi_q(S(\boldsymbol{\pi}_{1:i-1})))}. \tag{18}$$

Since the conditional probability of the mutation order does not depend on mutation times $\boldsymbol{u}$, then the marginal probability of the mutation order $\Pr(\pi_{1:n};\boldsymbol{\theta},h_0)$ is also equal to (18). □

Appendix B: Pre-processing data

Here we discuss additional details on how we preprocessed the data.

B.1. Ends of B-cell receptor sequences

If we are interested in modeling the effect of k-mer motifs on the hazard rate where $k > 1$, then the positions at the ends of the sequences must be properly handled. The issue is that the positions at the ends might not have enough neighboring nucleotides to fully construct a k-mer motif.

In order to deal with this issue, we first preprocess our data by trimming the two ends of the BCR sequences until the ends of the naive and mutated sequences are the same. We then assume that these end positions are fixed and not part of the mutation process.

For example, if we are interested in modeling how 3-mer motifs affect the mutation rate of the center position, we need to handle the special case of the two positions at the ends of the sequence. Given a naive BCR sequence and its associated mutated sequence, we trim away the positions at the ends of both sequences until the first and last positions are the same. If our trimmed sequence is of length p', we suppose that only positions 2 through $p' - 1$ can undergo mutation. We can now apply our estimation method since all positions use the same feature vector mapping.

Appendix C: Other simulations

C.1. Reconstructing mutation history

Since most clonal families contain multiple sequences, including all sequences without reconstructing the shared mutation history within each family can introduce bias by considering some mutations more than once. To overcome this bias, we consider two approaches: we can either attempt to estimate this history using standard methods, or we can randomly sample a single sequence from each clonal family. For the former case, to date, there are no methods that incorporate context-specific mutation models; we introduce one standard and useful approach to consider for a single clonal family.

Assume we have a collection of nucleotide sequences that have mutated away from a known naive sequence. In time, as we mutate away from this naive sequence, a series of intermediate nucleotide sequences are introduced on the way to obtaining the mutated sequences. These intermediate sequences, known as "ancestral states," are related to one another and to our observed sequences by an unknown phylogeny: a tree of dependencies that ties all sequences together by common ancestry. For a comprehensive treatment of phylogenetics, see Felsenstein (2003).

Unfortunately we do not observe these ancestral states. A simple approach to estimate them is to use parsimony imputation (Farris, 1970), a method that

TABLE 3
Statistics on reconstructing $\boldsymbol{\theta}$ using various data preprocessing methods.

Data processing	Model	Relative $\boldsymbol{\theta}$ error	Kendall's tau
All data	`samm`	0.486	0.650
	`SHazaM`	0.655	0.576
Imputation	`samm`	0.508	0.629
	`SHazaM`	0.700	0.557
Sampling	`samm`	0.484	0.655
	`SHazaM`	0.700	0.525

minimizes the total number of mutations that occur on the tree. Reconstructing ancestral states using parsimony with `dnapars` (Felsenstein, 2005) involves searching through a number of candidate trees and computing the minimum number of changes necessary to obtain each tree. Among the equally parsimonious trees returned by `dnapars`, we choose the first one to compute mutation contexts.

To determine the optimal data processing strategy between sampling, imputing ancestral states, and including all sequences without imputation, we simulate 3000 clonal families with realistic sizes. The composition for each of these clonal families is determined by sampling at random a cluster size and an inferred naive sequence from the `partis`-processed Cui et al. (2016) dataset. Cluster sizes range from 1–109. The median cluster size is two, and about 40% of all clusters are singletons. Sequences are 2.5% mutated on average. We take $\boldsymbol{\theta}$ to be a random resampling of $\boldsymbol{\theta}_{\texttt{MK}}$ parameters (Cui et al., 2016).

In Table 3 we see imputing ancestors using parsimony does not provide any improvements in the model fit in most cases. Given that mutations in the simulation above occur based on the sequence context, the relatively poor performance of imputing ancestors may be due to the heterogeneity of mutation rates among sites (Ho and Jermiin, 2004). Sampling a random descendant from each clonal family decreases the relative error for `samm`. For `SHazaM`, using all of the data results in the lowest relative error, most likely due to the fact that `SHazaM` fits mutabilities differently when not enough observations are present, and this case has more data than in the case of sampling. In Section 4, we sample from each clonal family to estimate the fit for `samm` while using all of the data for `SHazaM`.

C.2. Model misspecification: mutating with replacement

Throughout this manuscript, we have assumed that the positions in a BCR sequence mutate at most once. This assumption is for computational simplicity: if a position can mutate more than once, our estimation procedure must consider every single possible nucleotide sequence. However, this may not be realistic biologically. In this section, we present a simulation study to see how `samm`'s accuracy changes when positions are allowed to mutate multiple times.

Much of the simulation settings are similar to before. For the somatic hypermutation model, we resample from $\boldsymbol{\theta}_{\texttt{MK}}$ – defined in Section 2.4 – for each

TABLE 4
Results on ten replicates of simulated data with standard errors (SE).

Mutation Rate (%)	True Model	Relative $\boldsymbol{\theta}$ error (SE)	Kendall's tau (SE)
1–5	Mutate at most once	0.395 (0.020)	0.738 (0.015)
	Mutate multiple times	0.339 (0.022)	0.737 (0.018)
5–15	Mutate at most once	0.182 (0.017)	0.811 (0.012)
	Mutate multiple times	0.203 (0.020)	0.796 (0.008)

3-mer motif, then randomly set half of them to zero. Each dataset consists of 300 simulated BCR sequences from a single mouse. Mutations are simulated using a survival model where each position can mutate multiple times versus at most one time. This simulation study is run ten times.

For low mutation rates of 1–5%, we have similar accuracy, or perhaps even better accuracy, when the model is misspecified (Table 4). The accuracy of the method is similar since a position is very unlikely to mutate more than once in a low mutation rate setting.

We also try higher mutation rates as it is common to see mutation rates of 5–15% in humans, especially in individuals with chronic viral infections (He et al., 2014). One would suspect that for higher mutation rates, our simplifying assumption would result in lower performance; Table 4 shows that we indeed have lower accuracy when the mutation rate is between 5 and 15%. However, the performance is not much worse than when our model is correctly specified. These results suggest that our simplifying assumption gives up only a small degree of computational accuracy for a huge gain in computational efficiency.