Title
Automating Soil Taxonomy

Author
Olds, Marc Austin

Publication Date
2022

Peer reviewed|Thesis/dissertation

Automating Soil Taxonomy

By

MARC A. OLDS

THESIS

Submitted in partial satisfaction of the requirements for the degree of

MASTER OF SCIENCE

in

Soils & Biogeochemistry

in the

OFFICE OF GRADUATE STUDIES

of the

UNIVERSITY OF CALIFORNIA

DAVIS

Approved:

Anthony T. O'Geen, Chair

Rebecca A. Lybrand

William R. Horwath

Committee in Charge

2022

Abstract

Soil Taxonomy has proven to be a valuable tool for communicating soils information. Since its inception in 1975, soils have been classified by manually working through criteria established in the Keys to Soil Taxonomy. Navigating this system is often time consuming, prone to error, and requires significant pedologic knowledge and training. Despite recent advancements in soil data storage and analysis, no method currently exists to automatically calculate soil taxa. Here, a new suite of computer algorithms is presented to aid in the determination of soil classification. This tool is designed to take in soil lab and field observations, calculate diagnostic features, and output a final classification to the suborder level for all orders and great group level for Gelisols, Histosols, and Alfisols. Demonstrations prove that this tool can reach classifications that match human-verified results. However, the tool's accuracy is highly dependent on the quality and completeness of input data. The automated process requires a massive collection of data and observations for every possible classification outcome. Thus, the tool inherently lacks the human/pedologist expertise that would allow a user to simplify the classification process and reduce data demand by only considering regionally relevant taxa. However, the tool can be constrained by users to better mimic expert behavior. This tool shows incredible potential for reducing the burden of updating outdated pedon taxonomy in NRCS databases by processing large amounts of data quickly and accurately. This tool also provides a computer resource that will help make Soil Taxonomy more accessible.

Acknowledgments

This work would not have been possible without Dylan Beaudette and his team. Thank you for all you have done to make soils information technology what it is today. Much of this project literally could not exist if not for your dedication and commitment.

I would have never pursued this path without the exceptional instruction and guidance from Randy Southard and Toby O'Geen. You ignited my passion for soils all those years ago and changed the course of my life. Emily Mixer shared in that passion and was with me from those first bold steps of this journey. I cannot express just how much your friendship has meant to me these last few years.

Steve Deverel and my colleagues at HydroFocus have supported me every step of the way. Thank you for everything you have done to help me grow and push myself further.

To my family, I appreciate everything you have done, and do, to help me through whatever life throws. Your love and support mean so much to me and I am lucky to have you. You are an odd bunch, but I love you.

Finally, my wonderful and loving partner, Melissa. There are few certainties in my life, but I am certain I would have never made it this far without you by my side. Thank you.

Table of Contents

List of Figures and Tables

Introduction

Soil Taxonomy (ST) was originally established as a tool to communicate information via the National Cooperative Soil Survey. The hierarchical nature of ST was developed to support different scales of mapping products. It has since evolved as a method of communication within the soil science discipline and helps to explain characteristics of the whole soil and relationships with soil forming factors (Soil Survey Staff, 1999). Since the adoption of this taxonomic system, the process of classifying soils has been performed manually by surveyors using available field descriptions and laboratory analysis. Accurate classification of soils is key to assessing the spatial relationships of soil map units and providing insight into land use, soil formational history, and soil management.

Soil classification is currently determined by applying measured and observed soil properties to a set of criteria outlined in the "Keys to Soil Taxonomy" (Keys) publication (Soil Survey Staff, 2014). The Keys is composed of a set of requirements that guides users through the hierarchy of classes: order, suborder, great group, subgroup, and family. The structure and conditional logic of the entries are periodically updated as the classification system is altered to best adapt to new information and pedologic models. Thus, NCSS is challenged with the need to reclassify a large archive of existing pedons to maintain continuity of soil survey information.

Navigating Soil Taxonomy and the Keys is challenging without familiarity with pedologic concepts and takes a great deal of time to complete, even for experts. The language and terminology used within the Keys may as well be an entirely different language to outsiders. Additionally, the National Resource Conservation Service (NRCS) maintains a massive digital repository of soils information, but there is currently no way to directly utilize these data to automatically generate soil taxa. Updates to soil survey have been hampered as Soil Taxonomy

has changed over time due to the effort needed to apply the new standards to an expansive collection of established soils.

Here, we have built the Soil Taxonomy R package that contains the logic of the "Keys to Soil Taxonomy" and standardizes the classification methodology through machine algorithms. New data structures have been developed to contain specific data and diagnostic attributes associated with soil taxa. This new tool will assist with the classification of soil data in bulk and is capable of interfacing with existing computer-based applications and data systems. Existing classifications can be reprocessed and updated with ease to reflect changes to available data and taxonomic standards.

Background

Soil Taxonomy

The USDA soil taxonomic system was first published in *Soil Taxonomy* in 1975 (Soil Survey Staff, 1975) and a second edition was published in 1999 (Soil Survey Staff, 1999). ST is organized in a hierarchy of classes based on their properties (Soil Survey Staff, 1999). The development of the taxonomic system focused primarily on the relationship between soil characteristics and soil forming factors. The taxa created from these classes proved vital to soil survey mappers responsible for delineating map unit boundaries (Soil Survey Staff, 1999). The highest levels of the classification hierarchy, soil orders, are rooted in the soil forming factors (Bockheim et al., 2014). Orders represent broad, dominant "themes" of soils. Within each order, suborders are often differentiated by climatic factors. The suborders are further subdivided into great groups and great groups are subdivided into subgroups. The taxa within each successive

level are intended to show the relationships and differentiate between soils of the parent level (Smith, 1986).

To classify a soil, one first uses the collected soil properties to determine the epipedon and additional subsurface diagnostic characteristics, such as argillic horizons. The process for walking through taxa is stepwise, beginning with the 12 orders. Each order has 3-8 suborders, each suborder has many great groups, and each great groups has many subgroups. At each taxa level, one proceeds down the sequence of taxa listed in the Keys until the criteria are met. Then, the next list within the selected taxa is checked.

Past Attempts

There have been attempts to build computer-based systems for ST. Fisher & Balachandran (1989) developed a system that led users down a procedural logic tree to determine classification to the order level. This system required users to provide yes/no answers to questions formed from ST Keys criteria and assumed that all diagnostic features had been predetermined. Buol & Rebertus (1991) developed a similar system a short time later. Galbraith (1996) and Galbraith et al. (1998b) prototyped an expert system that also estimated diagnostic features and attempted to account for specific ST Keys criteria. This system employed estimations and expert alternatives to address missing data. The tool successfully determined diagnostically relevant features and classified accurately Histosols, Spodosols, Andisols, and Oxisols. This prototype also identified and made use of tabular datasets. The soil taxonomy tool discussed in this paper is most similar to the system developed by Galbraith and others (1998b) but greatly expanded to include all soil orders, all suborders, and many great groups.

Recent Work

Recent developments in computerized systems for soil science have contributed greatly to the suite of resources available for accessing and processing soils data. The creation of Web Soil Survey (Soil Survey Staff, 2005) provided digital access to soil survey maps and spatial data. SoilWeb (Beaudette & O'Geen, 2009) built on the success of Web Soil Survey and also further streamlined the process of viewing spatial soils data. SoilWeb expanded soil digital products to provide depth plots of key soil properties, direct links to pedon data, and smoother integration with other digital tools such as Google Earth. Open-source R programming developments in Algorithms for Quantitative Pedology (AQP) (Beaudette et al., 2013), SoilDB (Beaudette, et al., 2013), and SharpshootR (Beaudette et al., 2013), have continued to explore modern methods of accessing soil data systems and providing tools for pedologic research. All these tools have standardized soil data analysis, widened accessibility to National Cooperative Soil Survey (NCSS) data, and established a framework for mass data processing. The SoilTaxonomy R package and classification tool would likely not have been possible without this past work.

Tool Design

The ST tool has been developed as part of the SoilTaxonomy R package which contains data and functions relevant to the maintenance, development, and application. The tool attempts to replicate the classification process as outlined in the Keys to Soil Taxonomy 12th Edition (Soil Survey Staff, 2014) and the code was written based on the written language currently within this edition of the Keys. The tool logic is structured to follow the same diagnostic and classification procedures prescribed by the Keys just as professional pedologists have historically described and classified soils. However, the Keys were not written with computer programming

in mind. The wording and specifications in Keys criteria had to be interpreted and translated into a format that would work within a framework of R programming while preserving the original intent and meaning of the written Keys.

The tool was designed such that the criteria for each taxa and diagnostic feature are contained within individual functions that are called upon procedurally. Two wrapper functions, *calc_diagnostics* and *calc_taxonomy*, house the specific logic for the individual functions as dictated by the requirements of the Keys. Diagnostic features are processed and calculated first, followed by hierarchical ST levels. The diagram in Figure 1 details an overview of the tool process.

By default, the tool employs a "blind" approach to the classification process. This refers to a process wherein the diagnostic characteristics of any given pedon are unknown, so all diagnostic features must be checked. This contrasts with an "expert" approach where the classifier only checks for diagnostic features that are relevant given the location, climate, formational history, and other characteristics informed by experience. Because some diagnostic features are dependent on the presence or absence of others, a specific procedural order has been developed for calculating diagnostic horizons based on the requirements of the Keys. The diagram in Figure 2 shows the current logic used for the blind approach. Given the modular nature of the tool's design, options were included for users to call a specific subset of diagnostic functions or taxa, instead of relying on the "blind" approach.

An important feature of the tool is its ability to track the results of the classification process. This information allows a user to understand the reasons for a particular end classification and various intermediate levels within ST (diagnostic features, hierarchical levels). This information is linked to a condition code corresponding to the line in the Keys with the

written narrative for the criteria. As the code proceeds through each diagnostic and taxa function, results (TRUE/FALSE) of the individual conditions are recorded in the *diags* or *tax* data structures and may then be reported as part of the "classification report" for the pedon. Recording these results helps to create more transparency of the tool's processes and demystify the classification system.

Data Structures

The tool employs three internal data structures: *hzs*, *diags*, and *tax*. All these data structures are linked by unique pedon IDs. The *hzs* data structure is a dataframe that contains all - the quantitative lab and field data for each horizon in each pedon. This data structure is required for all diagnostic and taxonomy functions. A user may directly supply all the data in this data structure. If data is missing for certain properties, the tool will attempt to make reasonable assumptions or calculations based on which data is available. 107 input variables were identified at the horizon level that are directly required by the Keys or were created to help break down other variables and aid in checking Keys criteria (Table 1). Data may be entered into the tool manually through a custom data structure (e.g., Excel spreadsheet) or through existing R data query services. Many soil properties are recorded within the *fetchKSSL* data structures from National Soil Information System (NASIS) and Kellogg Soil Survey Laboratory (KSSL). The *fetchKSSL* function is part of the soilDB R package. It was developed to download KSSL data into an R environment for further analysis. Table 1 outlines each soil property used by the tool as well as any assumptions/calculations used to fill gaps in the data.

The *diags* data structure is a list of dataframes that contains all the diagnostic information for the pedon (e.g., epipedons, subsurface horizons, moisture regime, etc.). This structure is required for all functions employed by the tool to calculate diagnostic features and taxonomy.

The structure is initialized by the user by passing a list of pedon IDs, soil moisture regimes, and soil temperature regimes. In each diagnostic dataframe, each row represents a pedon. The columns in each diagnostic dataframe vary between features but all have a logical column to indicate if the feature is present for the pedon. For applicable features, there are columns for top and bottom depth, feature thickness, and each condition checked for the feature. When calculating diagnostic features, the *diags* structure is updated in a diagnostic function and returned as an output.

The *tax* data structure is a list of dataframes that contains all the taxonomy information for the pedon. This structure is required for taxonomy functions. The structure is initiated by the user passing a list of pedon IDs. Similar to the *diags* structure, each dataframe represents a taxon and contains logical columns for recording if the taxa conditions are met and the results of each checked condition. This structure is updated and returned as an output for each function.

Table 1. *Horizon input data structure variables, corresponding columns from fetchKSSL, required units, and input assumptions when data is not provided. Where applicable, the assumption may be an equation to calculate the value from other given properties. Otherwise, the value in the column is used to fill in blank spaces in the input table.*

Horizon property	**fetchKSSL horizon property**	**Data Type**	**Property**	**Units In Keys**	**Units in NASIS/KSSL**	**Assumptions / Calc. If Not Given**	**Notes / Comments**
aadipyridyl	-	boolean	flag for reaction to alpha,alphadipyridal	boolean	NA	FALSE	
acec7		float	apparent CEC, pH7	cmol/kg	cmol/kg	=cec7/%clay	
aecec	-	float	apparent effective CEC (8.2)	cmol/kg	cmol/kg	=ecec/%clay	
albic_vol	-	float	albic material volume	%	NA	0	

aloxalate	al_ox	float	ammonium oxalate extractable aluminum	%	%	0	
al_kcl	al_kcl	float	kcl extractable aluminum	cmol/kg	cmol/kg	0	
andic	-	boolean	flag for andic properties	boolean	NA	FALSE	
anhyd	-	float	% anhydrite	%	NA	0	
anthraquiccond	-	boolean	flag for anthraquic conditions	boolean	NA	FALSE	
aquic	-	boolean	flag for aquic conditions	boolean	NA	FALSE	
art	-	boolean	presence of artifacts	boolean	NA	FALSE	determined from ‘u’ in hzname
bioturb_vol	-	float	bioturbation volume	%	NA	0	
bs7	bs7	float	base saturation at pH 7	%	%		
bs82	bs82	float	base saturation at pH 8.2	%	%		
caco3	caco3	float	calcium carbonate content %	%	%	0	
cec7	cec7	float	CEC at pH 7	cmol/kg	cmol/kg		
cec82	cec82	float	CEC at pH 8.2	cmol/kg	cmol/kg		
cemagt	-	string	cementing agent		NA	NA	determined from hzname; ‘m’ and ‘k’/’kk’; ‘q’; ‘yy’; ‘s’; ‘kq’; ‘z’
cemented	-	boolean	flag for if hz is cemented	boolean	NA	FALSE	determined from ‘m’ in hzname
cindfragpum	-	boolean	flag for cindery, fragmental, or pumacious material	boolean	NA	FALSE	

clay_films	-	boolean	flag for if clay films are present	boolean	NA	FALSE	
claysizedcarb	co3_cly	float	clay sized carbonates %	%	%	0	
claytotal	clay	float	% total clay	%	%		
color_d		string	overall color - dry	field book	NA		
color_w		string	overall color - wet	field book	NA		
colorchroma_d	colorchoroma | d_chroma	int	dry color chroma	field book	field book		estimated from overall color if moist/dry given
colorchroma_w	colorchoroma | m_chroma	int	wet color chroma	field book	field book		estimated from overall color if moist/dry given
colorhue_d	colorhue | d_hue	string	dry color hue	field book	field book		estimated from overall color if moist/dry given
colorhue_w	colorhue | m_hue	string	wet color hue	field book	field book		estimated from overall color if moist/dry given
colorvalue_d	colorvalue | d_value	int	dry color value	field book	field book		estimated from overall color if moist/dry given
colorvalue_w	colorvalue | m_value	int	wet color value	field book	field book		estimated from overall color if moist/dry given
consec_sat	-	integer	consecutive days saturated in a year	-	NA	0	

cryoturb	-	boolean	flag for evidence of cryoturbation	boolean	NA	FALSE	determined from ‘jj’ in hzname
cumusat	-	integer	cumulative days saturated in a year	-	NA	0	
dbovendry	db_od	float	oven dried bulk density	g/cm3	g/cm3		
dbthirdbar	db_13b	float	bulk density at 1/3 bar (33 kPa)	g/cm3	g/cm3		
densic	-	boolean	flag for densic contact/layer	boolean	NA	FALSE	determined from ‘Cd’ hzname
desgnmaster	-	string	master hz designation	field book	field book		determined from hzname
ec	ec_12pre * 2	float	EC	dS/m	dS/m	0	
ec_15	-	float	EC 1:5 soil:water supernatant	dS/m	NA	0	
ecec	ecec	float	effective CEC at pH8.2 (CEC 8.2 + KCl ex. Al)	cmol/kg		=cec82 + al_kcl	
effclass	-	string	effervescence class	field book	NA	noneffervescent	if ‘kk’ in hzname: strongly effervescent; if ‘kq’: slightly effervescent
esp	-	float	ESP	%	%	0	estimate from sar
ex_ca	ex_ca	float	exchangeable Ca	cmol/kg	cmol/kg	0	
ex_mg	ex_mg	float	exchangeable Mg	cmol/kg	cmol/kg	0	
ex_na	ex_na	float	exchangeable Na	cmol/kg	cmol/kg	0	
extracid	acid_tea	float	extractable acid	cmol/kg	cmol/kg	0	
failure	-	string	failure class	field book	NA	brittle	

feoxalate	fe_ox	float	ammonium oxalate extractable iron	%	%		
feoxide	iron_oxide_total	float	iron oxide	% wt	% wt	0	
fragtotvol	frags	float	rock fragment total volume	% vol	% wt	0	
gelic_mat	-	boolean	flag for gelic materials	boolean	NA	FALSE	
gypsum	gypl20	float	% weight of gypsum	%	%	0	
humanTpAlt	-	boolean	flag for human transported/altered material	boolean	NA	FALSE	determined from '^' in hzname
hzdepb	hzn_top	float	hz bottom depth in cm	cm	cm		
hzdept	hzn_bot	float	hz top depth in cm	cm	cm		
hzname	hzn_desgn	string	hz name	field book	field book		
hzthk	-	float	hz thickness	cm	cm		can be calculated from hzdept and hzdepb
ksat	Ks	float	saturated hydraulic conductivity in um/sec	um/sec	log(cm/d)		
lamellae_thk	-	float	lamellae thickness	cm	NA	0	
lamellae_vol	-	float	lamellae volume	%	NA	0	
lithdisc	-	boolean	flag for lithologic discontinuity	boolean	NA	FALSE	can be determined from hzname
lithic	-	boolean	flag for lithic contact	boolean	NA	FALSE	can be determine from 'R' in hzname
melanic_idx	-	float	melanic index	-	NA	0	
midden	-	boolean	flag for midden	boolean	NA	FALSE	

moss_vol	-	float	volume of moss fibers	%	NA	0	
n	-	float	n value	integer	NA	1	calculated
om	estimated_om	float	organic matter %	%	%	0	
oc	oc	float	organic carbon %	%	%	0	
p_ret	p_nz	float	phosphorus retention	%	%	0	
paralithic	-	boolean	flag for paralithic contact	boolean	NA	FALSE	can be determined from ‘Cr’ in hzname
pedon_id	pedon_key	string	unique pedon ID	-	-		
permafrost	-	boolean	flag for permafrost	boolean	NA	FALSE	can be determined from ‘f’ or ‘ff’ in hzname
ph1to1h2o	ph_h2o	float	pH in 1:1 water	pH	pH		
ph_cacl2	ph_cacl2	float	pH in CaCl2	pH	pH		
plinthite_vol	-	float	volume of plinthite	%	NA	0	can be determined from ‘v’ in hzname
redoxcolor	-	string	overall color of redox concentrations	field book	NA	NA	
redoxcolorchroma	-	int	chroma of redox concentrations	field book	NA	NA	
redoxcolorhue	-	string	hue of redox concentrations	field book	NA	NA	
redoxcolorvalue	-	int	value of redox concentrations	field book	NA	NA	
redoxconcen	-	boolean	flag for redox concentrations	boolean	NA	FALSE	if concretions or masses given, set to TRUE
redoxconcre	-	boolean	flag for redox concretion	boolean	NA	FALSE	
redoxdepl	-	boolean	flag for redox depletions	boolean	NA	FALSE	

redoxmass	-	boolean	flag for redox masses	boolean	NA	FALSE	
rll	-	boolean	flag for root limiting layer	boolean	NA	FALSE	duripan, fragipan, petrocalcic, petrogypsic, placic, densic, lithic, paralithic, petroferric
rrc	-	string	rupture resistance class		NA		
sandtotal	sand	float	total sand %	%	%		
sar	sar	float	SAR	unitless/%	unitless/%	13 if 'n'; else 0	calculate: ex_na / (0.5*(ex_ca+ex_mg))
sat_mech	-	string	describes the mechanisms of wetting	-	NA	NA	"EPISAT" or "ENDOSAT”
seccarb	-	float	% secondary carbonates	%	NA	0	
silt	silt	float	% silt	%	%	=100-sand-clay	
siltco	-	float	coarse silt %	%	%	0	
slake_h2o	-	float	volume that slakes in water	%	NA	100	determined based on cemented and cemagt
slake_hcl	-	float	volume that slakes in HCl	%	NA	100	determined based on cemented and cemagt
slake_koh	-	float	volume that slakes in KOH	%	NA	100	determined based on cemented and cemagt
slake_naoh	-	float	volume that slakes in NaOH	%	NA	100	determined based on cemented and cemagt
sphgm_vol	-	float	sphagnum volume as %	%	NA	0	

spodic_mat_pct	-	float	% spodic material in hz	%	NA	0	
structgrade	structgrade	string	structure grade	field book	field book	structureless	structureless if structtype is massive
structsize	structsize	string	structure size (of peds)	field book	field book	NA	
structtype	structtype	string	aggregate size	field book	field book	massive	massive if structgrade is structureless
sulfidic_mat	-	boolean	sulfidic material	boolean	NA	FALSE	
texture	lab_texture_class	string	texture class without modifier	field book	field book	NA	can be calculated given sand and clay
texturemod	-	string	texture modifier	field book	NA	NA	
vglass	-	float	volume of volcanic glass	%	%	0	
whc	whc	float	water holding capacity			NA	used to calculate n
wormhole_vol	-	float	wormhole volume	%	NA	0	only used for agric horizon

Figure 1. *Diagram of the overall soil taxonomy tool process showing the relationship of the three main data structures. Hzs (dataframe), Diags (list), and Tax (list) represent the internal data structures of the tool. The names listed within each box are the elements of each data structure. In Hzs, these are columns of the table. For Daigs and Tax, these are dataframes within the list. The blue boxes (check_inputs, calc_diagnostics, and calc_taxonomy) are functions. Arrows leading to the functions indicate data passed as inputs. Arrows leading out of the functions indicate outputs.*

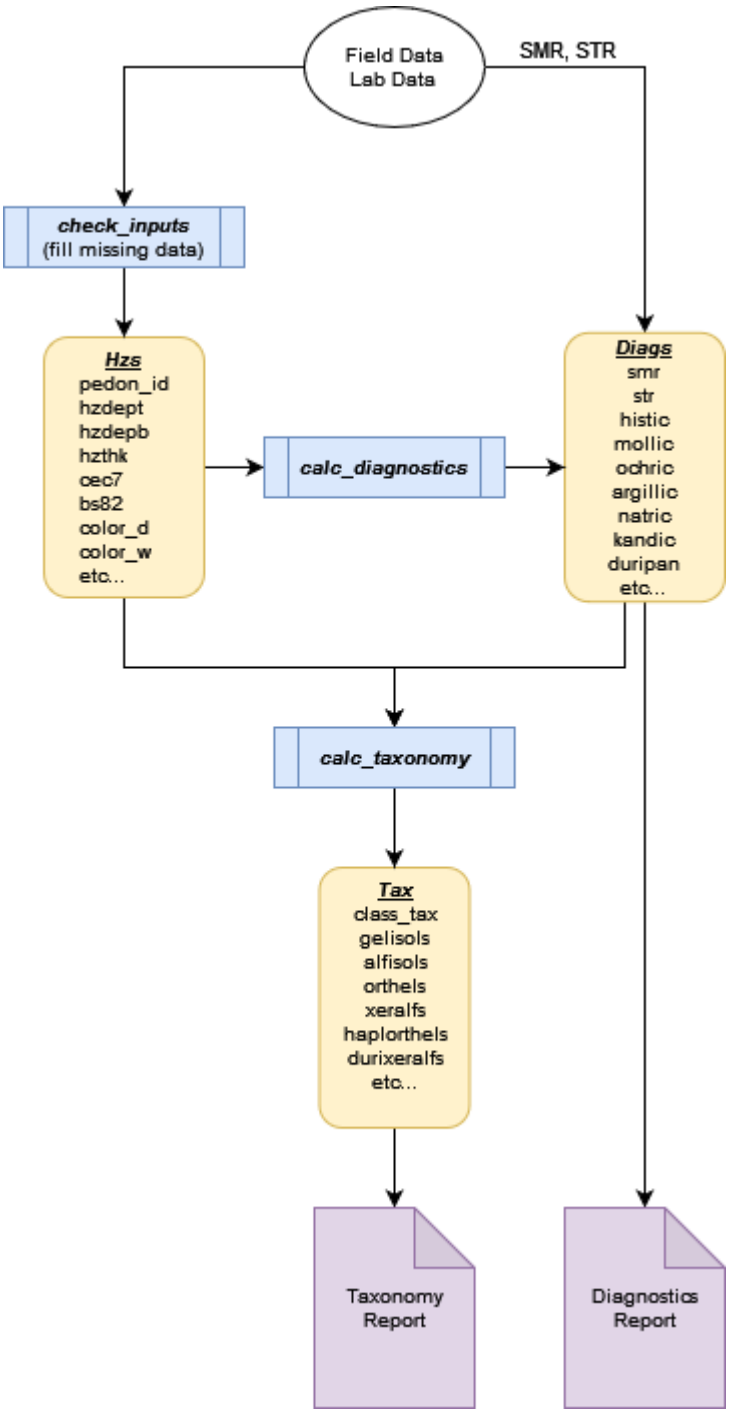
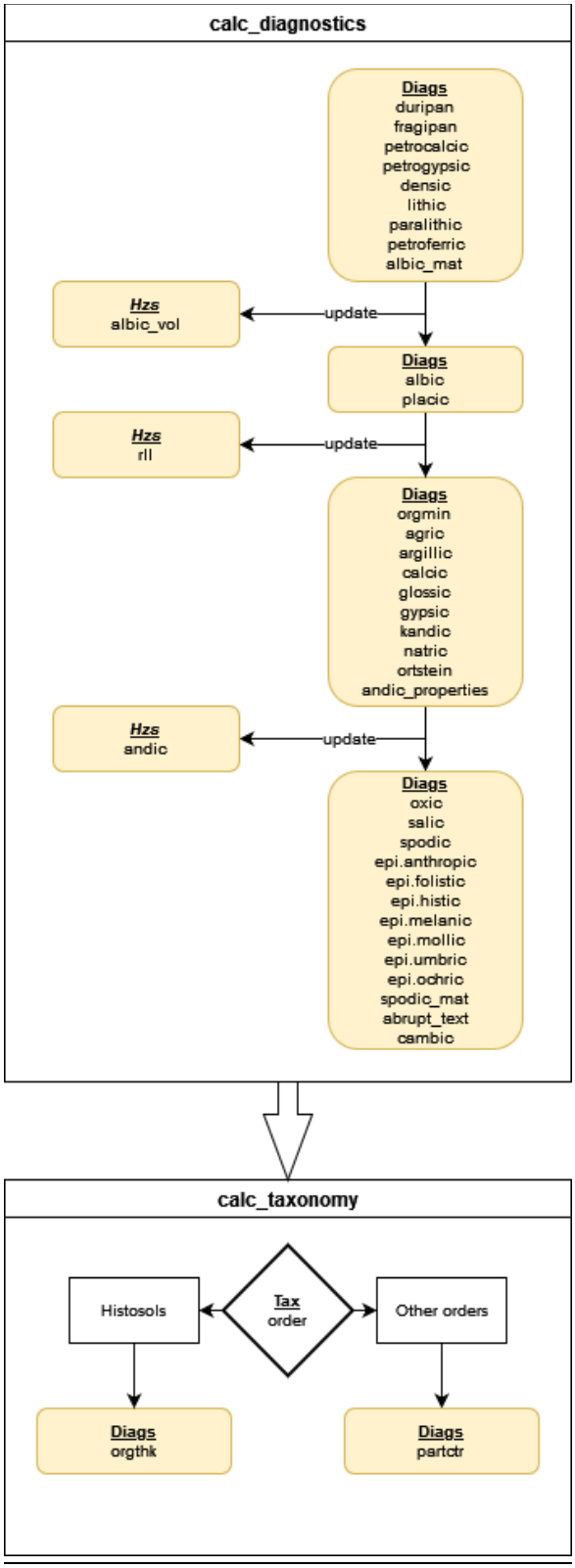


Figure 2. *Diagram showing the logic for calculating diagnostic features.*



Testing the tool

Two demonstrations are discussed to assess the accuracy and functionality of the taxonomy tool. In the first demonstration, input datasets were manually curated with the intention of classifying each of the twelve soil orders. Classifications were calculated for the order and suborder level for twelve input pedons. Classification was carried out to the great group level for Gelisol, Histosol, and Alfisol pedons as the code for all great groups within these orders had been completed at the time of writing. To build the inputs, available data from a specific soil series were used as a starting template, pulling from a combination of data sources including NASIS/KSSL, official series descriptions, and SSURGO. When input data for certain properties were unavailable, reasonable values were generated that would fall within the range of the target classification. The purpose of this demonstration was to correct errors in the code and prove the tool's functionality given a complete dataset.

In the second demonstration, the tool's accuracy was evaluated based on actual pedon data and their corresponding classifications by soil survey staff. Pedon data was collected through the *fetchKSSL* function within the soilDB R package, which queries the KSSL snapshot hosted within SoilWeb. Color data was also retrieved from NASIS through the *fetchKSSL* function. Queries were performed by soil series name. Each soil series used for this demonstration were selected based on the availability and completeness of pedon data, regional significance and mapped extent, and diversity of classification. For this demonstration, emphasis was placed on alfisols given their significance and abundance in California and elsewhere in the continental United States. Results were filtered to remove shallow pedons and pedons with invalid boundaries. Pedons missing data for clay or sand content, moist and dry color, base saturation at pH 8.2, and CEC at pH 7 were also excluded from the analysis. Missing color data

was estimated from the moist or dry counterpart using the *estimateSoilColor* function in the aqp R package. Rupture resistance class, which is not provided by *fetchKSSL*, was estimated from SSURGO for each correlated soil series. The input soil moisture and temperature regimes were estimated from the official series descriptions. These complete pedons served as input data for the soil taxonomy tool and calculated the classification to the suborder level. All Alfisols up to the great group level were classified. The calculated classifications were then compared with the classification provided for each pedon from the NRCS database. As the provided classifications may be outdated, the tool could only be assessed to see if it calculated the same or different classification as is currently listed by NRCS. Matching classifications do not necessarily indicate a "correct" classification nor did mismatching classification necessarily indicate "incorrect" classifications. Each taxa level was assessed independently as lower taxa are dependent on higher levels. After determining the matching and mismatching classifications, the pedon data of each pedon with mismatches was reviewed at each level and categorized the most probable reason for the mismatch. 5 categories of mismatching classifications were determined: 1. incomplete data [ID] (of properties recorded in NASIS/KSSL); 2. missing data (properties required by the Keys were not recorded in NASIS/KSSL) & incomplete data [MD/ID]; 3. incorrect or old classifications [IC]; 4. incorrect or old classifications & incomplete data [IC/ID]; and, 5. issues with translating or interpreting Keys logic into the tool code [LIC].

The input data used for the first demonstration is provided in the R code included in Appendix A (A-26 through A-47). The data is in dataframe format assigned to the variable *demo_1_hz*. The code used to query and filter the NRCS database for the second demonstration is provided in Appendix A (A-17) in the variable *m* from the function call of *merge_kssl_series*.

Results

Demonstration #1 - Curated Inputs

The tool performed very well when given complete datasets tailored to the Keys' requirements. In the first demonstration, the tool successfully matched all target orders, suborders, and great groups. Table 2 shows which suborders and great groups were tested for this demonstration and the resulting classification from the taxonomy tool.

***Table 2**. Target and calculated classifications for manually curated input data.*

Pedon ID	Target Order	Target Suborder	Target Great Group	Calculated Order	Calculated Suborder	Calculated Great Group
gel_1	Gelisols	Turbels	Aquiturbels	Gelisols	Turbels	Aquiturbels
hist_1	Histosols	Saprists	Haplosaprists	Histosols	Saprists	Haplosaprists
spod_1	Spodosols	Orthods	-	Spodosols	Orthods	-
and_1	Andisols	Xerands	-	Andisols	Xerands	-
ox_1	Oxisols	Torrox	-	Oxisols	Torrox	-
vert_1	Vertisols	Uderts	-	Vertisols	Uderts	-
arid_1	Aridisols	Argids	-	Aridisols	Argids	-
ult_1	Ultisols	Udults	-	Ultisols	Udults	-
moll_1	Mollisols	Ustolls	-	Mollisols	Ustolls	-
alf_1	Alfisols	Xeralfs	Haploxeralfs	Alfisols	Xeralfs	Haploxeralfs
incept_1	Inceptisols	Ustepts	-	Inceptisols	Ustepts	-
ent_1	Entisols	Orthents	-	Entisols	Orthents	-

Demonstration #2 - NASIS/KSSL Inputs

After filtering pedons for the most complete datasets, 125 pedons remained to be processed through the taxonomy tool. Notably, 10 of these pedons did not have a given classification as provided by *fetchKSSL*, so it was not possible to determine a matching or mismatching classification. Of the remaining 115 pedons, the tool calculated the same order as the given classification for 103 pedons (89.6%). Figure 3 shows the taxonomic distribution of the matching and mismatching pedons at the order level. The number of mismatches tended to increase with increasing numbers of or pedons per order. However, some orders had no mismatches irrespective of the number of representative pedons tested (e.g. Andisols (4), Histosols (1) Ultisols (12), Vertisols (1).

Since subsequent levels of classification are dependent on higher levels (e.g. order), only pedons with matching orders could be adequately compared at the suborder level and matching suborders compared at the great group level. Of the 103 pedons with matching orders, the tool calculated the same suborder as the provided classification for 86 pedons (83.5% of the matching orders). Figure 4 details the taxonomic distribution at the suborder level. Most mismatches were associated with taxa with aquic conditions.

Alfisols

Of the original 115 pedons without missing classifications, there were 58 pedons identified as Alfisols from *fetchKSSL*. The tool determined 53 (91.4%) of these pedons to be Alfisols, matching the given classification. Of these, the tool also matched 47 pedons (88.7% of the matching Alfisols) with the same suborder and a matching great group for 30 pedons (63.8% of the matching suborders). The tool classified 51.7% of pedons as the same great group as the originally provided great group classification. Figure 5 shows the Alfisols great group

distribution for matching and mismatching classifications. No clear trend was observed among mismatches at the great group level, except perhaps that no mismatches occurred for Alfisols with duripans or fragipans.

Figure 6 breaks down the proposed reasons for all 46 mismatching classifications at all levels. The most mismatches (20) can be attributed to incomplete data from *fetchKSSL* followed closely by incorrect/old classifications (19). We attribute 5 mismatches to a combination of both missing data and incomplete data. 1 mismatch was likely due to both incorrect/old classification and incomplete data. 1 mismatch likely occurred due to issues interpreting the logic of the Keys and its translation into the code.

Figure 3*. Taxonomic distribution of matching and mismatching classifications at the order level. "Matching Orders" refers to the number of pedons correctly matched for that order. "Different Order" refers to the number of pedons calculated as an order different than the bin.*

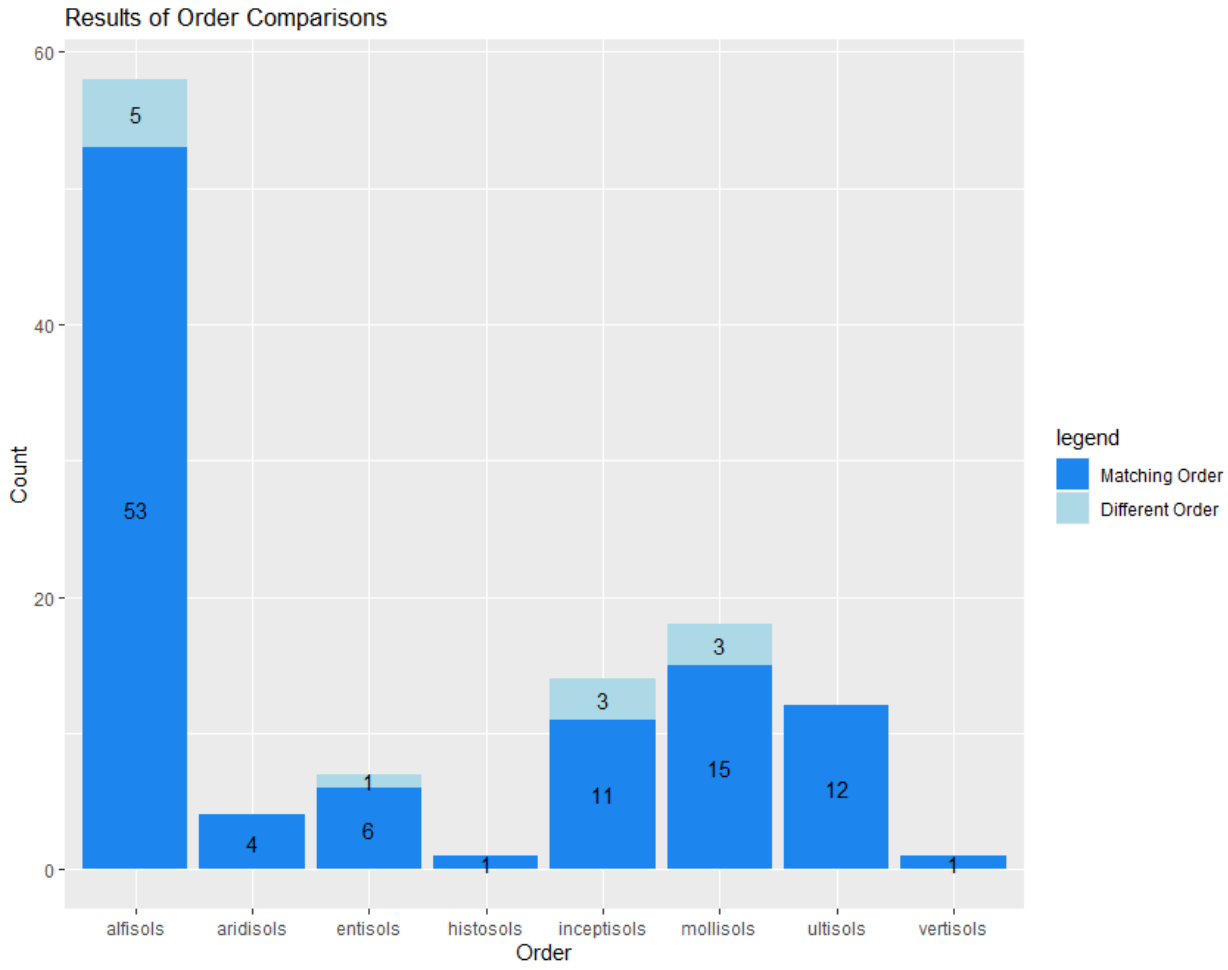


Figure 4. *Taxonomic distribution of matching and mismatching classifications at the suborder level.*

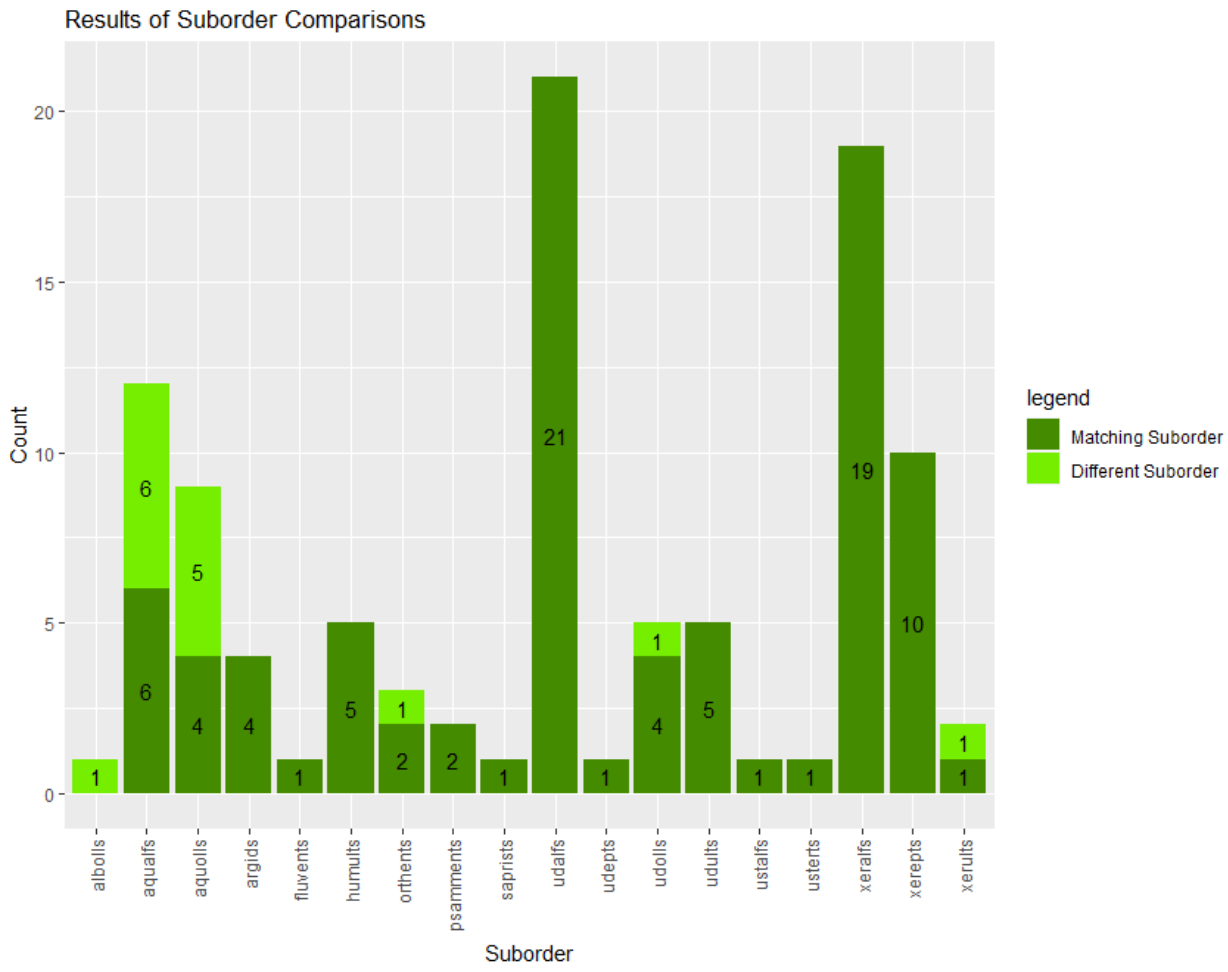


Figure 5. *Taxonomic distribution of matching and mismatching classifications at the great group level for pedons originally classified as Alfisols.*

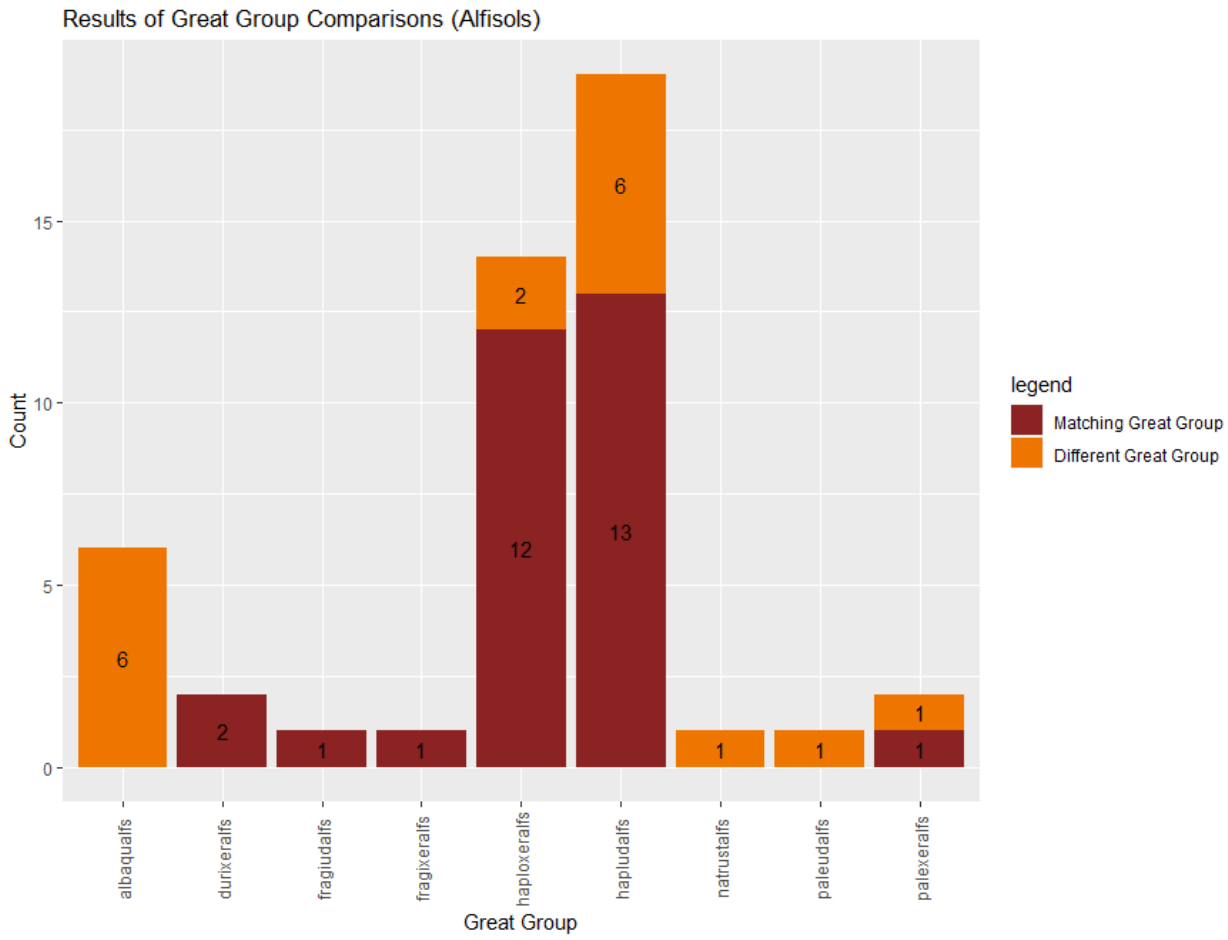


Figure 6. *Frequency and categories of all mismatching classifications for all levels.*

IC = Incorrect/Old Classification

ID = Incomplete Data

MD/ID = Missing Data & Incomplete Data

IC/ID = Incorrect/Old Classification & Incomplete Data

LIC = Logic/Interpretation/Code

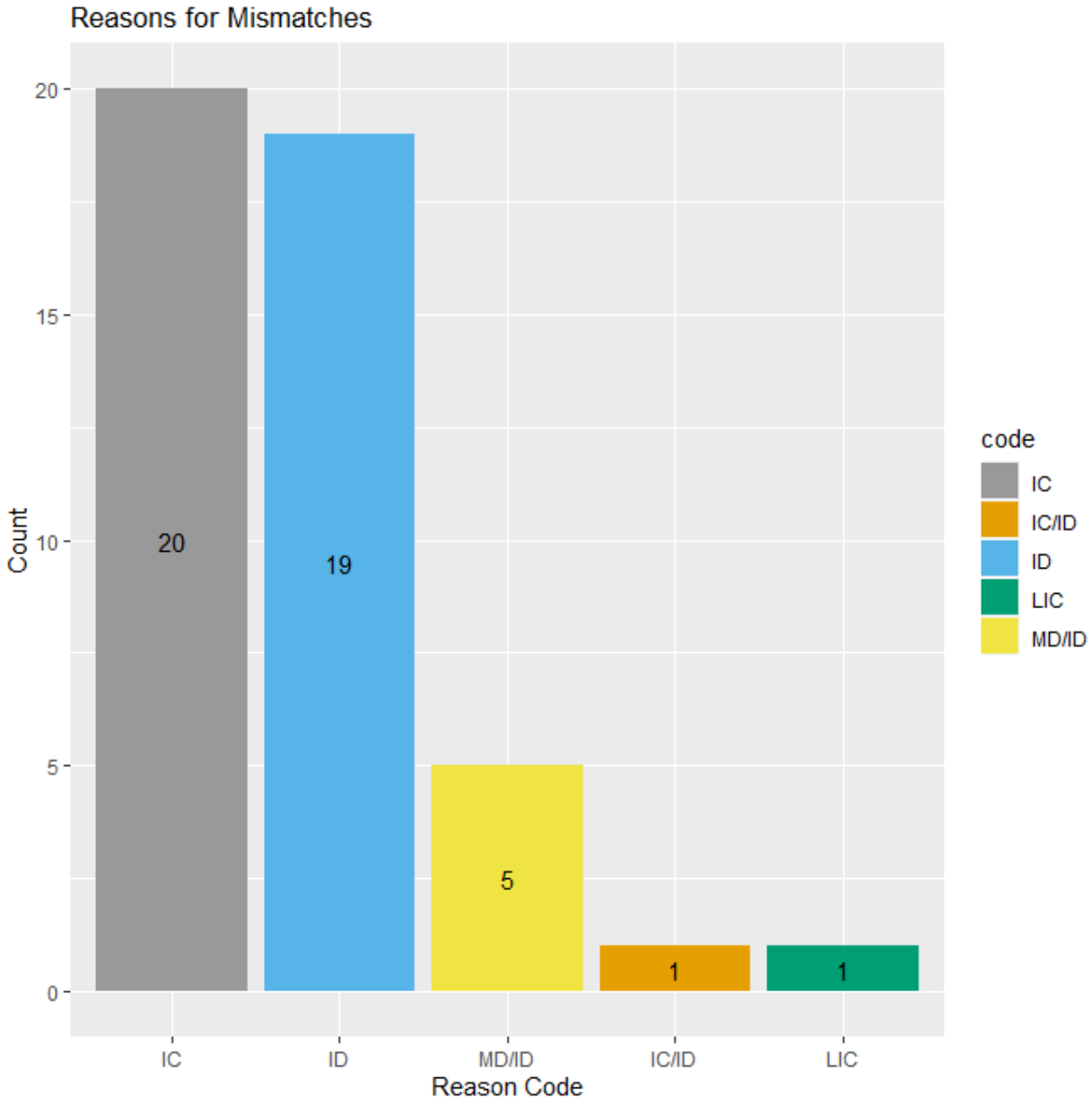


Figure 7. *Breakdown of mismatches across taxa levels.*

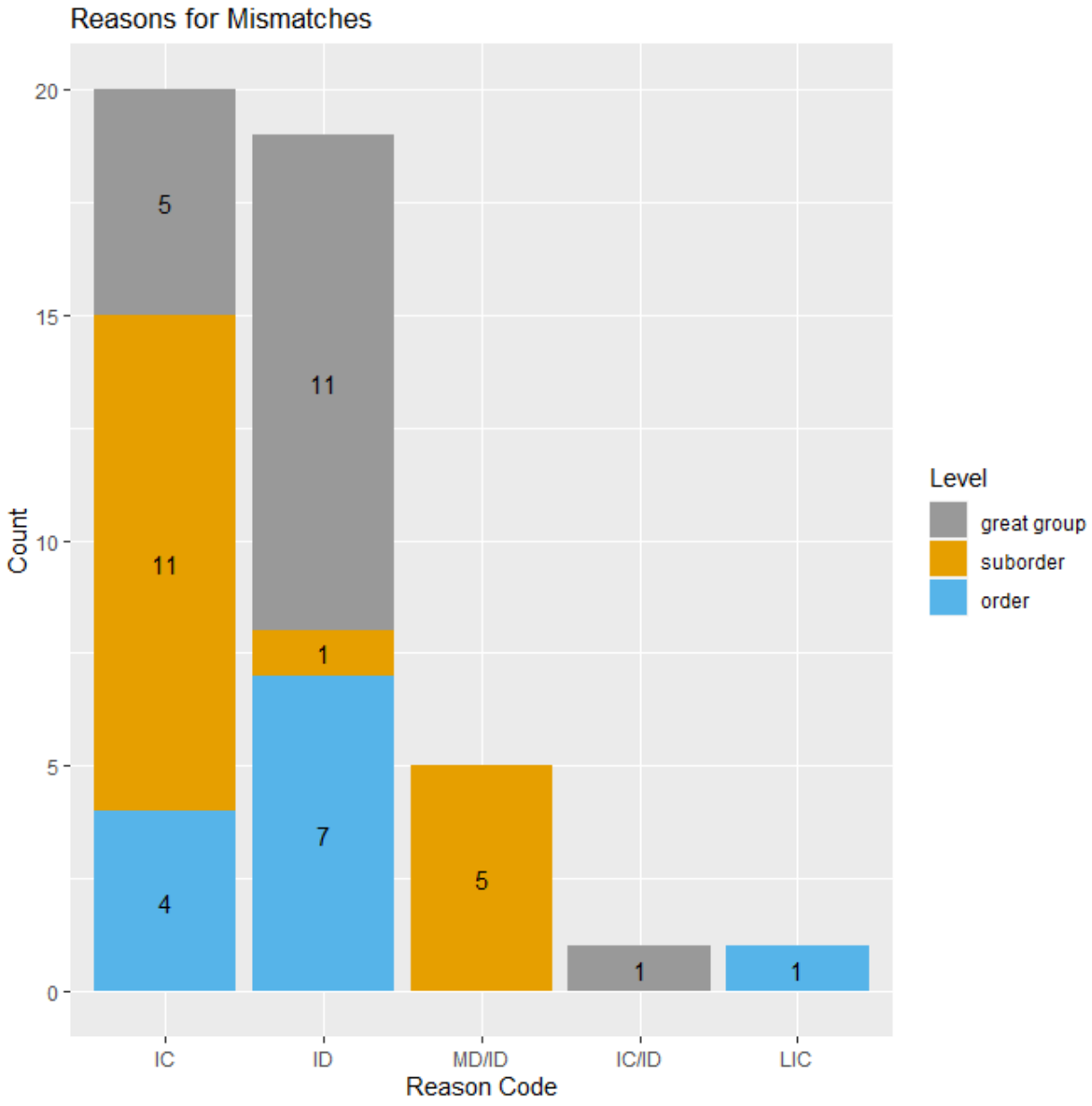


Table 3. *Summary of all mismatches. The reported classification is the classification listed in the NRCS database for each pedon which may differ from the current classification for the series.*

pedon id	Series	Taxa Level	Reported Classification	Calculated Classification	Reason
6992	auberry	order	alfisols	mollisols	ID
6364	cajon	order	entisols	aridisols	IC
15097	clarion	order	mollisols	inceptisols	ID
15111	clarion	order	inceptisols	mollisols	ID
15112	clarion	order	alfisols	mollisols	ID
15114	clarion	order	mollisols	inceptisols	ID
11642	hotaw	order	alfisols	ultisols	IC
45786	lindley	order	alfisols	mollisols	ID
55455	otwood	order	alfisols	ultisols	IC
10155	pentz	order	mollisols	inceptisols	ID
34051	whiterock	order	inceptisols	entisols	IC
40884	zook	order	inceptisols	entisols	LIC
11580	cherokee	suborder	aqulfs	udalfs	MD/ID
21272	cherokee	suborder	aqualfs	udalfs	MD/ID
43101	cherokee	suborder	aqualfs	udalfs	ID
1970	drummer	suborder	aquolls	udolls	IC
2642	drummer	suborder	aquolls	udolls	IC
2794	handford	suborder	orthents	fluvents	IC
14433	lethent	suborder	argids	cambids	IC
14436	lethent	suborder	argids	cambids	IC
11639	mariposa	suborder	xerults	humults	IC
7084	tranquilar	suborder	udolls	ustolls	IC
7091	tranquilar	suborder	albolls	ustolls	IC
17162	wynoose	suborder	aqualfs	udalfs	MD/ID
17165	wynoose	suborder	aqualfs	udalfs	MD/ID
18537	wynoose	suborder	aqualfs	udalfs	MD/ID
1473	zook	suborder	aquolls	udolls	IC
22888	zook	suborder	aquolls	udolls	IC
23554	zook	suborder	aquolls	udolls	IC

17845	argonaut	great group	palexeralfs	haploxeralfs	IC
20514	cherokee	great group	albaqualfs	endoaqualfs	IC
42416	cherokee	great group	albaqualfs	endoaqualfs	ID
48357	cherokee	great group	albaqualfs	endoaqualfs	ID
48363	cherokee	great group	albaqualfs	endoaqualfs	ID
48484	cherokee	great group	hapludalfs	paleudalfs	ID
48529	cherokee	great group	albaqualfs	haploaqaulfs	ID
7319	corning	great group	haploxeralfs	palexeralfs	IC
7319	corning	great group	haploxeralfs	palexeralfs	IC
3224	holland	great group	haploxeralfs	palexeralfs	IC
54931	lowell	great group	hapludalfs	paleudalfs	ID
55099	lowell	great group	hapludalfs	paleudalfs	ID
68852	lowell	great group	hapludalfs	paleudalfs	ID
48656	marion	great group	hapludalfs	paleudalfs	ID
16879	reedsburg	great group	paleudalfs	haplustalfs	ID
24540	swanboy	great group	natrustalfs	haplustalfs	ID
18529	wynoose	great group	albaqualfs	endoaqualfs	IC/ID

Discussion

The overall performance of the tool may vary greatly depending on individual user goals and intentions. Two primary categories of users may use this tool: 1. researchers and individuals experimenting with taxonomy; and 2. NRCS staff working to update and maintain classifications. The first group would generally use the tool to learn more about taxonomy and as an aid for their research. It is expected that this group would be more concerned with the taxonomy system itself, working with estimates of soil properties (supplemented with real data) and experimenting with different combinations of data. In contrast, NRCS would be looking for definitive results based on real data collected from the lab and field and associated reasoning why/how classifications are reinterpreted or changed.

Mismatching Classifications

Figure 7 provides a breakdown of each taxonomic level within the mismatch categories. In general, mismatches were spread across the major mismatch categories at all taxonomic levels and there were no clear trends across taxa given this subset of pedons used in the analysis. The two biggest categories, IC and ID, would likely continue to be the primary reasons for mismatches when using this tool to analyze pedons from the NRCS database. It is important to note that ID mismatches may shift into IC if the datasets were more complete and continued to support a different classification. But until more data is provided, there is no certainty that the mismatch accurately reflects a classification consistent with current requirements.

A comparison of the existing classification and the new tool classification is provided in Table 3. The mismatches at the order level appear to be taxa that a pedon could reasonably jump between given the right variation in keys properties. For example, the mismatches between Alfisols and Ultisols (pedons 11642 and 55455) were entirely due to the value of base saturation at pH 8.2 at the depth required by the Keys (i.e. a 35% threshold at 2-m depth for Alfisols vs Ultisols). Not all pedons simply jumped down to the next taxa in the sequence but were classified as an order several taxa further down the list. Many of the calculated Mollisols were likely classified as such due to the dry colors meeting the requirements for a mollic epipedon, but in many pedons, dry color had to be estimated from moist color. It should be noted that ST is designed to find the taxa that reflect the properties observed in the database for any given pedon. Some adjacent taxa may reflect gradation of certain properties and diagnostics where values just below the threshold fall within criteria for strict the next listed taxa. However, this does not always occur, and the classification will simply jump to the next taxa where the conditions are satisfied.

At the suborder level, most taxa are governed by moisture regime or temperature regime, relatively static properties determined by location. Pedons will usually fall within one suborder without any concern for fluctuation or gradation in value. Most of the mismatches occurred in Aqualfs and Aquolls which are both suborders that have more complex requirements than most other suborders. The strict redox requirements in these taxa were a common reason for an ID or MD/ID mismatch in these suborders, data commonly missing from NRCS databases.

At the great group level, many ID mismatches were due to estimated colors or missing Ksat values in the case of Albaqualfs. There was enough complete data for 5 pedons to determine that the calculated classifications were accurate.

The one pedon placed in the LIC category contained buried horizons. Currently, the tool is unable to handle buried horizons as they influence classification, so this mismatch could be attributed to issues with the code of the tool.

Quality and Availability of Input Data

The Keys often requires more data than may be available in NRCS databases for a specific pedon. For example, the Keys specifically requires certain volumes of spodic materials, albic materials, and plinthite, but information for these properties is often missing in regions where the features simply do not exist. While other data may help point to the presence or absence of these properties, the actual volume cannot be accurately determined without direct measurement. Thus, reasonable assumptions must be made for these variables and usually consist of a binary choice: 0% if the property is not present or 100% if the property is present. Other properties, such as the size of dry cracks (necessary for Vertisols), are excluded from the tool's code entirely as the data may not even be recorded in the field given the dynamic nature of these features.

Incomplete data in the NRCS databases and the absence of certain properties altogether proved to be one of the greatest challenges in evaluating the tool. In addition to the properties already mentioned, NASIS and KSSL also do not record data such as saturation frequency (cumulative or consecutive), the mechanism for saturation (epi vs endo), failure class, moss volume, n-value, rupture resistance class/consistence (often recorded but not available through *fetchKSSL*), and sulfidic materials. Many more of the variables listed in Table 1 are also not recorded, but some properties can be reasonably calculated or determined from other data in the databases, provided that the data is available. For example, apparent CEC (pH 7) can be calculated as CEC7 divided by percent clay. Thus, this property does not need to be provided by the user so long as CEC7 and clay content are available for the calculation. Other such assumptions, estimates, or calculations made by the tool are detailed in Table 1. If the datasets are incomplete (the property is recorded in the database, but there is no data for a particular pedon or horizon), multiple dependent properties may not be determined, and the classification would be severely limited. Considering the use of the tool by researchers and individuals, one can ensure that all relevant data is properly filled in or reasonably estimated (as shown in Demonstration #1). However, NRCS staff would likely rely on whatever is provided in NASIS and KSSL (as in Demonstration #2). It was found that databases were often missing data for bulk density (especially for 'O' horizons), CEC, base saturation at pH 8.2, and OM / OC content. Color was also a common missing property and, if available, only the dry or moist color would be provided. Crucial properties such as clay and sand content were also missing from some pedons. During the analysis, many pedons had to be excluded due to incomplete datasets as the tool simply cannot proceed without many of these properties. It proved difficult to find suitable datasets to effectively demonstrate the tool's performance. Even of the pedons that survived the

filtering process, more than 40% of mismatching classifications could be attributed to incomplete data.

Expert vs. Blind Classification

The Keys to Soil Taxonomy was originally written and designed to be traversed by a human mind. A field pedologist tends to only focus soil properties and data relevant to the classification of the soil at hand and often assesses multiple diagnostic conditions concurrently. The structure of the Keys, and the classification system as a whole, partly depends on human reasoning to make sensible interpretations. A human user is capable of resolving issues such as circular logic that may arise within the language of the Keys as they apply to unique circumstances. A pedologist's experience and familiarity with the classification process has allowed the Keys to be an effective tool for soil professionals. The soil taxonomy tool, as it may be employed by less experienced users, must check every diagnostic feature procedurally through the *blind* classification process. However, this process increases the minimum amount of complete data required for the tool to function accurately, adding complexity and growing the potential for error. Users may attempt to emulate the human component of the classification process by specifying a subset of checks to perform, but this would require some degree of expertise with soil classification to be effective. Galbraith et al. (1998a, 1998b) identified this kind of expert knowledge as a fundamental component of any automatic classification system. While the tool may not fit their description of an expert system perfectly, it is believed this approach (e.g. user options, reasonable assumptions, etc.) is sufficient for accomplishing the goals of soil taxonomy.

Interpreting the Keys

The language of the Keys can, at times, be challenging to effectively replicate in R programming. One of the key goals when constructing the taxonomy tool was to address barriers between inexperienced users and the taxonomy system. This often meant that a balance had to be struck between code complexity and user experience. As backend tool complexity increases, it becomes more difficult to troubleshoot unexpected results and link the programming back to the written Keys. Conversely, simplified programming may shift this burden to the user by requiring many input parameters. Still, other issues arose from the Keys logic itself that proved troublesome to incorporate into the code at all.

One such issue that was encountered was the occurrence of circular logic. For example, here is an excerpt from condition 3 in the requirements for a cambic horizon:

"*3. Has properties that do not meet the requirements for an anthropic, histic, folistic, melanic, mollic, plaggen, or umbric epipedon, a duripan or fragipan, or an argillic, calcic, gypsic, natric, oxic, petrocalcic, petrogypsic, placic, salic, spodic, or sulfuric horizon; and […]* " (Soil Survey Staff, 2014 p. 13)

The cambic horizon must not meet the requirements of the diagnostic features listed condition 3, so this would call for the logic to determine that none of these features are also present within the same part of the profile as the cambic. However, the mollic and umbric epipedons may also depend on the presence of a cambic horizon to determine their bottom depth:

Mollic Epipedon condition 6a(3)b:

"*(b) The lower boundary of the deepest of an argillic, cambic, natric, oxic, or spodic horizon; or […]*" (Soil Survey Staff, 2014 p. 9)

Mollic Epipedon condition 6c(2):

"*(2) The lower boundary of the deepest of an argillic, cambic, natric, oxic, or spodic horizon; or […]*" (Soil Survey Staff, 2014 p. 9)

Umbric Epipedon condition 6a(3)b:

"*(b) The lower boundary of the deepest of an argillic, cambic, natric, oxic, or spodic horizon; or […]*" (Soil Survey Staff, 2014 p. 11)

Umbric Epipedon condition 6c(2):

"*(2) The lower boundary of the deepest of an argillic, cambic, natric, oxic, or spodic horizon; or […]*" (Soil Survey Staff, 2014 p. 11)

While one can procedurally check for each of the other diagnostic features of condition 3 before checking for the cambic horizon, one would be unable to perform the checks for a mollic or umbric epipedon without knowing if a cambic horizon is present. While attempts could be made to resolve this issue with more advanced programming techniques, the code behind the tool would ultimately begin to become very complex and the problem may still not be completely solved through the code alone. The current iteration of the tool alters the mollic and umbric requirements by removing the check for the cambic horizon. The cambic horizon requirements were also simplified to check for horizons of alteration that do not overlap with other diagnostic

features. While this may lead to inaccurate results, it is not believed this portion of the tool impacted either of the analyses.

A similar case can be seen in the requirements for spodic materials:

Spodic Materials conditions 2b(3) and 2b(4)

"*(3) Al plus 1/2 Fe percentages (by ammonium oxalate) totaling 0.50 or more, and half that amount or less in an overlying umbric (or subhorizon of an umbric) epipedon, ochric epipedon, or albic horizon; or*

(4) An optical density of oxalate extract (ODOE) value of 0.25 or more, and a value half as high or lower in an overlying umbric (or subhorizon of an umbric) epipedon, ochric epipedon, or albic horizon." (Soil Survey Staff, 2014 p. 22)

In this case, umbric and ochric epipedons depend on a spodic horizon (which is composed of spodic materials).

Umbric Epipedon condition 6a(3)b

"*(b) The lower boundary of the deepest of an argillic, cambic, natric, oxic, or spodic horizon; or […]*" (Soil Survey Staff, 2014 p. 11)

Umbric Epipedon condition 6c(2):

"*(2) The lower boundary of the deepest of an argillic, cambic, natric, oxic, or spodic horizon; or […]*" (Soil Survey Staff, 2014 p. 11)

Ochric Epipedon

"*The ochric epipedon includes eluvial horizons that are at or near the soil surface, and it extends to the first underlying diagnostic illuvial horizon (defined below as an argillic, kandic, natric, or spodic horizon). [...]*" (Soil Survey Staff, 2014 p. 10)

While these features are not directly dependent on spodic material as the mollic and umbric were for the cambic horizon, this circular logic means that calculated spodic materials cannot be used to determine the presence of a spodic horizon. The spodic horizon itself is not dependent on the epipedon, so it may be processed higher in the logic precedence and before the epipedons. To solve this problem, a user input was created for spodic materials to help determine spodic horizons.

Here another example with the Keys entry for Torrerts:

"*FD. Other Vertisols that, if not irrigated during the year, have cracks in normal years that remain closed for less than 60 consecutive days during a period when the soil temperature at a depth of 50 cm from the soil surface is higher than 8 ^{o}C.*" (Soil Survey Staff, 2014 p. 305)

This entry is asking for a lot of information. Realistically, the only way to perform this check would be to create several user inputs. However, additional targeted inputs may only be used for a handful of entries (or even just this single entry) and increases the input burden for the user. In this case, a reasonable interpretation could be to check for the implied soil moisture regime (aridic) from the suborder formative factor, '*torr*-'. Complicated entries with niche data requirements are not uncommon in the Keys. When necessary, reasonable liberties must be taken when translating and interpreting the language of the Keys for a computerized framework to

maintain usability (by both experts and inexperienced users) while preserving the general functionality and intent of soil taxonomy. Alternatively, these complications as depicted by the tool could help identify future revisions to ST that may be necessary.

Applications

Once completed, the soil taxonomy tool will have much to offer to government agencies, researchers, and the general public as an accessible tool for navigating the soil classification system. The NRCS, which maintains soil taxonomy standards, is responsible for updating all pedon classifications in their database to adhere to the current version of the Keys. This task has historically been performed manually by NRCS staff and can be quite time consuming considering there are over 22,000 soil series and many more components that would need to be evaluated. From the analysis, it was determined that 20 (43%) mismatching classifications could be attributed to incorrect or old classifications, so it is clear that many pedons still need to be updated. Each of these pedons were reviewed and it was confirmed that the data did not support the listed classification based on the current Soil Taxonomy standards. The tool may provide a means to speed up classification updates by processing pedon data in bulk from existing databases using a method similar to the demonstration. This process would be limited to pedons with sufficiently complete datasets, but it could still alleviate some of the pressure on the NRCS backlog.

The tool also has potential to aid geoscience research by providing a means to quickly estimate diagnostic features and soil classification. The hope is that newcomers to pedology will feel less intimidated by Soil Taxonomy and will choose to incorporate more soil taxonomy terminology into their own work. Expanded use of this system could improve communication across disciplines and with stakeholders. Many fields already use R tools to assist with

geoscience work, so another R package may fit neatly into existing workflows. The tool may also begin to open new doors for pedologists to explore Soil Taxonomy itself. For example, the tool may be used to quickly assess how diagnostic features and classification change when provided a range of soil characteristics. One could change a major input property, such as clay, and immediately observe how a different value propagates through the calculated diagnostics and final classification. A finely incremented array of values for the property could be run in a matter of minutes. Additionally, the tool could also be used to model the distribution of soil properties, such as carbon, as a function of their taxonomy. The tool can account for all possible pathways to a classification, so the internal composite boundaries for a property within a given taxon could be quantified as well as the specific conditions that define those boundaries. Coupled with other data within the SoilTaxonomy R package, pedologists may also now have new opportunities to explore the language and logic of the Keys in ways they could not before.

The development of this tool adds yet another resource for soil science public engagement. The author envisions that this tool will be the beginning of a new suite of educational resources focused on ST. Planned expansions include easy to use web interfaces for running the tool as well as definitions of soil properties and explanations for how they are used in ST. Users will also be provided an end report from the tool to help explain the reasoning behind the calculated taxonomy. There is hope that these additions will help users from all backgrounds gain a better understanding and appreciation of ST.

Next Steps

At this current stage of development, the soil taxonomy tool can calculate almost all possible diagnostic features and classify taxa to the suborder level for all orders and to the great group level for Gelisols, Histosols, and Alfisols. Continued development of the tool should finish

the great groups and begin adding subgroups. Eventually, family classification could be added to the tool's capabilities. A web interface is currently in development to make the tool more accessible for users that do not have experience with R. The website will also include educational resources to help explain soil properties and the taxonomy process as a whole. We also envision that the SoilTaxonomy R package will be connected with AQP and soilDB to build an interconnected library of R utilities for soil science. If adopted, the soil taxonomy tool may also encourage improvements to existing NRCS databases to better serve the requirements of soil taxonomy.

Concluding Thoughts

It has been demonstrated that the logic outlined in the Keys to Soil Taxonomy can be effectively replicated in a computerized system through R code. This new method for traversing the Keys can quickly process bulk inputs of soil properties to calculate soil diagnostic features and taxa. The overall performance and accuracy of the tool are dependent on the quality of the input dataset, where more complete datasets will yield more accurate classifications. This tool will be useful for updating outdated pedon classifications and may also create new opportunities for soil education and research. ST serves an important role in modern pedology by providing a convenient system to explain soil properties and formation. ST is becoming ever more ubiquitous in the broader scientific community. Recent work, such as Stolt et al. (2021), has begun exploring changes to ST. This tool and the work presented here may aid similar inquiries and provoke new questions worthy of consideration. This work may be a first step towards modernizing ST to meet the needs of a new generation of soil science.

References

Beaudette, D. E., & O'Geen, A. T. (2009). Soil-Web: an online soil survey for California, Arizona, and Nevada. *Computers & Geosciences*, *35*(10), 2119-2128.

Beaudette, D. E., Roudier, P., & O'Geen, A. T. (2013). Algorithms for quantitative pedology: A toolkit for soil scientists. *Computers & Geosciences*, *52*, 258-268.

Beaudette, D. E., Skovlin, J., Roecker, S., & Beaudette, M. D. (2013). Package 'soilDB'.

Beaudette, M. D. (2013). Package 'sharpshootR'.

Bockheim, J. G., Gennadiyev, A. N., Hartemink, A. E., & Brevik, E. C. (2014). Soil-forming factors and Soil Taxonomy. *Geoderma*, *226*, 231-237.

Buol, S. W., and R. A. Rebertus. (1991). Soil taxonomy: Keys to classification. SMSS Computer Software Programs No. 1. USAID, USDA-NRCS, Washington, DC.

Fisher, P. F., & Balachandran, C. S. (1989). STAX: A Turbo Prolog rule-based system for soil taxonomy. *Computers & Geosciences*, *15*(3), 295-324.

Galbraith, J. M. (1996). *Soil TaxES-An expert system for soil taxonomy*. Agronomy Mimeo R96-2. Dept. Soil, Crop, Atmospheric Science, Cornell University, Ithaca, NY.

Galbraith, J. M., & Bryant, R. B. (1998a). A functional analysis of soil taxonomy in relation to expert system techniques. *Soil Science*, *163*(9), 739-747.

Galbraith, J. M., Bryant, R. B., & Ahrens, R. J. (1998b). An expert system for soil taxonomy. *Soil Science*, *163*(9), 748-758.

Smith, G. D. (1986). *The Guy Smith interviews: Rationale for concepts in soil taxonomy* (No. 11). Cornell University, Department of Agronomy.

Soil Survey Staff. (1975). *Soil taxonomy: a basic system of soil classification for making and interpreting soil surveys* (No. 436). US Government Print.

Soil Survey Staff. (1999). *Soil taxonomy: A basic system of soil classification for making and interpreting soil surveys*. 2nd edition. Natural Resources Conservation Service. U.S. Department of Agriculture Handbook 436.

Soil Survey Staff. (2005). Web Soil Survey. Natural Resources Conservation Service, United States Department of Agriculture.

Soil Survey Staff. (2014). Keys to Soil Taxonomy, 12th ed. USDA-Natural Resources Conservation Service, Washington, DC.

Stolt, M. H., O'Geen, A. T., Beaudette, D. E., Drohan, P. J., Galbraith, J. M., Lindbo, D. L., Monger, H. C., Needelman, B. A., Ransom, M. D., Rabenhorst, M. C., & Shaw, J. N. (2021). Changing the hierarchical placement of soil moisture regimes in Soil Taxonomy. *Soil Science Society of America Journal*, 85(3), 488– 500.

Appendices

Appendix A: analysis.R

Appendix B: calc_diagnostics.R

Appendix C: calc_taxonomy.R

Appendix D: check_inputs.R

Appendix E: diag_struct.R

Appendix F: tax_struct.R

Appendix G: epipedons.R

Appendix H: subsurface_diag.R

Appendix I: orders.R

Appendix J: suborders.R

Appendix K: gelisols.greatgroups.R

Appendix L: histosols.greatgroups.R

Appendix M: alfisols.greatgroups.R

Appendix N: helpers.R

Appendix O: ST_logic.rda

Appendix A:

analysis.R

```
library('dplyr')
library('soilDB')
library('aqp')
library('pbapply')
library('readxl')
library('SoilTaxonomy')
library('farver')
library('gower')
library('ggplot2')
library('reshape2')


## Load all ST functions ##

ff <- "C:/Users/Marc/Desktop/NRCS Project/soil_taxonomy/R/soiltaxonomy/R/main"
files.sources = list.files(ff)
sapply(files.sources, function(f){
  fp <- paste(ff,f,sep='/')
  print(fp)
  source(fp)
  })



##### Working with Series #####

s <- c("series",
       "Amador",
       "Angelscreek",
       "Argonaut",
       "Arol",
       "Auberry",
       "Auburn",
       "Beybek",
       "Bonanza",
       "Bottlehill",
       "Cajon",
       "Calflax",
       "Catlin",
       "Cecil",
       "Chaix",
       "Cherokee",
       "Clarendon",
       "Clarion",
       "Cometa",
       "Contra Costa",
       "Copperopolis",
       "Corning",
       "Crimeahouse",
       "Deerflat",
       "Delpiedra",
       "Devilsnose",
       "Donpedro",
       "Drummer",
```

```
"Dunstone",
"Exchequer",
"Fancher",
"Fiddletown",
"Flanly",
"Goldwall",
"Gopheridge",
"Hanford",
"Hetchy",
"Holland",
"Hotaw",
"Iron Mountain",
"Jennylind",
"Jocal",
"Lethent",
"Lilygap",
"Lindley",
"Loafercreek",
"Lowell",
"Luckymine",
"Mantree",
"Marion",
"Mariposa",
"Mccarthy",
"Miami",
"Mokelumne",
"Montpellier",
"Motherlode",
"Musick",
"Nebish",
"Nedsgulch",
"Neer",
"Otwood",
"Ospital",
"Pardee",
"Pescadero",
"Pentz",
"Ponto",
"Priestgrade",
"Reedsburg",
"Red bluff",
"Redapple",
"Redding",
"Rindge",
"San Joaquin",
"Santa",
"Sehorn",
"Sierra",
"Sites",
"Sixbit",
"Supan",
"Swanboy",
"Swissranch",
"Toomes",
```

```
        "Trabuco",
        "Tranquilar",
        "Wallyhill",
        "Wardsferry",
        "Whiterock",
        "Windthorst",
        "Wukusick",
        "Wynoose",
        "Zack",
        "Zook")
```

```
series_str <- data.frame(
  stringsAsFactors = FALSE,
                           series = c("Amador","Angelscreek","Argonaut","Arol",
                                      "Auberry","Auburn","Beybek","Bonanza",
                                      "Bottlehill","Cajon","Calflax","Catlin",
                                      "Cecil","Chaix","Cherokee","Clarendon",
                                      "Clarion","Cometa","Contra Costa",
                                      "Copperopolis","Corning","Crimeahouse",
                                      "Deerflat","Delpiedra","Devilsnose","Donpedro",
                                      "Drummer","Dunstone","Exchequer",
                                      "Fancher","Fiddletown","Flanly","Goldwall",
                                      "Gopheridge","Hanford","Hetchy","Holland",
                                      "Hotaw","Iron Mountain","Jennylind",
                                      "Jocal","Lethent","Lilygap","Lindley",
                                      "Loafercreek","Lowell","Luckymine","Mantree",
                                      "Marion","Mariposa","Mccarthy","Miami",
                                      "Mokelumne","Montpellier","Motherlode",
                                      "Musick","Nebish","Nedsgulch","Neer",
                                      "Otwood","Ospital","Pardee","Pescadero",
                                      "Pentz","Ponto","Priestgrade","Reedsburg",
                                      "Red bluff","Redapple","Redding",
                                      "Rindge","San Joaquin","Santa","Sehorn",
                                      "Sierra","Sites","Sixbit","Supan","Swanboy",
                                      "Swissranch","Toomes","Trabuco",
                                      "Tranquilar","Wallyhill","Wardsferry",
                                      "Whiterock","Windthorst","Wukusick","Wynoose",
                                      "Zack","Zook"),
                              str = c("thermic","thermic","thermic","thermic",
                                      "thermic","thermic","thermic","thermic",
                                      "mesic","thermic","thermic","mesic",
                                      "thermic","mesic","thermic","thermic",
                                      "mesic","thermic","thermic","thermic",
                                      "thermic","thermic","thermic","thermic",
                                      "mesic","thermic","mesic","thermic",
                                      "thermic","thermic","mesic","thermic","thermic",
                                      "thermic","thermic","thermic","mesic",
```

```
                                    "mesic","mesic","thermic","mesic",
                                    "thermic","mesic","mesic","thermic","mesic",
                                    "thermic","mesic","mesic","mesic",
                                    "mesic","mesic","thermic","thermic",
                                    "thermic","mesic","frigid","mesic","mesic",
                                    "mesic","thermic","thermic","thermic",
                                    "thermic","mesic","thermic","mesic",
                                    "thermic","mesic","thermic","thermic",
                                    "thermic","figid","thermic","thermic","mesic",
                                    "thermic","mesic","mesic","mesic",
                                    "thermic","thermic","frigid","mesic",
                                    "thermic","thermic","thermic","mesic",
                                    "mesic","thermic","mesic"),
                           smr = c("xeric","xeric","xeric","ustic","xeric",
                                    "xeric","xeric","xeric","xeric","aridic",
                                    "aridic","udic","udic","xeric",
                                    "aquic","udic","udic","xeric","xeric",
                                    "xeric","xeric","xeric","xeric","xeric",
                                    "xeric","xeric","aquic","xeric","xeric",
                                    "xeric","xeric","xeric","xeric","xeric",
                                    "xeric","xeric","xeric","xeric","xeric",
                                    "xeric","xeric","aridic","xeric",
                                    "udic","xeric","udic","xeric","xeric",
                                    "udic","xeric","xeric","udic","xeric",
                                    "xeric","xeric","xeric","udic","xeric",
                                    "xeric","udic","xeric","xeric","xeric",
                                    "xeric","xeric","xeric","udic","xeric",
                                    "xeric","xeric","aquic","xeric","xeric",
                                    "xeric","xeric","xeric","xeric",
                                    "xeric","ustic","xeric","xeric","xeric",
                                    "ustic","xeric","xeric","xeric","ustic",
                                    "xeric","aquic","ustic","aquic")
                )
```

```
## a <- soilDB::get_OSD(s)[,c('SERIES','TAXONOMIC.CLASS')] # outdated
# smrs <- c("aquic","aridic","ustic","udic","xeric")
# b <- soilDB::fetchOSD(s)

# a <- horizons(fetchKSSL(series="tanana"))

# TODO:
# - add andisols to the mix
# - record estimated colors
merge_kssl_series <- function(series) {

  res <- suppressMessages(soilDB::fetchKSSL(series=series,
                                            returnMorphologicData=TRUE,
                                            returnGeochemicalData=TRUE,
```

```
                                                simplifyColors=TRUE,
                                                progress=TRUE))

  res$SPC$taxonname <- tolower(res$SPC$taxonname)
  res$SPC <- HzDepthLogicSubset(res$SPC)

  res$SPC$nHorizons <- profileApply(res$SPC, function(i) {

    h <- horizons(i)
    idx <- grep('R|Cr|Cd|qm', h$hzn_desgn, invert = TRUE)

    return(nrow(h[idx, ]))
  })

  nHz.per.taxonname <- floor(tapply(res$SPC$nHorizons, res$SPC$taxonname,
median))
  res$SPC$expectedHorizons <- nHz.per.taxonname[match(res$SPC$taxonname,
names(nHz.per.taxonname))]

  res$SPC <- subset(res$SPC, (res$SPC$nHorizons / res$SPC$expectedHorizons) >=
1)

  .sub <- function(i) {

    # ensure that profile starts at 0cm
    .tophz <- horizons(i[, , .FIRST])
    .top <- .tophz[[horizonDepths(i)[1]]]

    if(.top > 0) {
      return(FALSE)
    }

    return(TRUE)
  }

  res$SPC <- subApply(res$SPC, .fun = .sub)

  # res <- aqp::subset(res, 'pedon_completeness_index' >= 4)
  # valid_depth <- aqp::checkHzDepthLogic(res$SPC)
  # res <- aqp::subset(res, valid_depth$valid)
  # res <- res[lengths(res)!=0]

  ## Filter for important data ##

  # need to filter everything again at the end

  spc <- res$SPC
  hzs <- horizons(spc)
  sites <- site(spc)

  hzs <- dplyr::filter(hzs,!grepl('UCD',labsampnum))
  sites <- dplyr::filter(sites,pedon_key %in% hzs$pedon_key)

  # colors <- dplyr::filter(res$morph$phcolor,labsampnum %in% hzs$labsampnum)
```

```
  frags <- dplyr::filter(res$morph$phfrags,labsampnum %in% hzs$labsampnum)
  struct <- dplyr::filter(res$morph$phstructure,labsampnum %in%
hzs$labsampnum)
  orig_diag <- dplyr::filter(res$morph$pediagfeatures, pedon_key %in%
hzs$pedon_key)
  redox <- dplyr::filter(res$morph$rmf, labsampnum %in% hzs$labsampnum)



  ### Explicit Fixes ###
  hzs$hzn_desgn[hzs$pedon_key=="2715"][c(3,4,5,6)] <-
c("Bt1","Bt2","Bt3","Bt4")




  ## Color ##

  idx_w_m <- is.na(hzs$m_hue) & is.na(hzs$m_value) & is.na(hzs$m_chroma) &
                  !is.na(hzs$d_hue) & !is.na(hzs$d_value) & !
is.na(hzs$d_chroma)

  idx_d_m <- is.na(hzs$d_hue) & is.na(hzs$d_value) & is.na(hzs$d_chroma) &
                  !is.na(hzs$m_hue) & !is.na(hzs$m_value) & !
is.na(hzs$m_chroma)

  m.moist <- aqp::estimateSoilColor(hzs$d_hue[idx_w_m],
                                    hzs$d_value[idx_w_m],
                                    hzs$d_chroma[idx_w_m],
                                    sourceMoistureState = 'dry')
  m.dry <- aqp::estimateSoilColor(hzs$m_hue[idx_d_m],
                                  hzs$m_value[idx_d_m],
                                  hzs$m_chroma[idx_d_m],
                                  sourceMoistureState = 'moist')


  sites$estWetColor[sites$pedon_key %in% unique(hzs$pedon_key[idx_w_m])] <-
TRUE
  sites$estDryColor[sites$pedon_key %in% unique(hzs$pedon_key[idx_d_m])] <-
TRUE


  hzs$m_hue[idx_w_m] <- m.moist$hue
  hzs$m_value[idx_w_m] <- m.moist$value
  hzs$m_chroma[idx_w_m] <- m.moist$chroma

  hzs$d_hue[idx_d_m] <- m.dry$hue
  hzs$d_value[idx_d_m] <- m.dry$value
  hzs$d_chroma[idx_d_m] <- m.dry$chroma

  idx_w <- !is.na(hzs$m_hue) & !is.na(hzs$m_value) & !is.na(hzs$m_chroma)
  idx_d <- !is.na(hzs$d_hue) & !is.na(hzs$d_value) & !is.na(hzs$d_chroma)
```

```
  hzs$color_w <- NA
  hzs$color_d <- NA

  hzs$color_w[idx_w] <- paste(hzs$m_hue[idx_w],
                              paste(hzs$m_value[idx_w],hzs$m_chroma[idx_w],sep='/'),sep='
')
  hzs$color_d[idx_d] <- paste(hzs$d_hue[idx_d],
                              paste(hzs$d_value[idx_d],hzs$d_chroma[idx_d],sep='/'),sep='
')



  # filter NAs

  # colors <- dplyr::filter(colors, !is.na(colorhue) & !is.na(colorvalue) & !
is.na(colorchroma) &
  #                           (colors$colormoiststst %in% c("Moist","Dry")))
  #
  # colors$colorhue <- trimws(colors$colorhue)
  # colors$colorvalue <- trimws(colors$colorvalue)
  # colors$colorchroma <- trimws(colors$colorchroma)
  #
  # idx_w <- colors$colormoiststst == "Moist" &
  #   !is.na(colors$colorhue) &
  #   !is.na(colors$colorvalue) &
  #   !is.na(colors$colorchroma)
  #
  # idx_d <- colors$colormoiststst == "Dry" &
  #   !is.na(colors$colorhue) &
  #   !is.na(colors$colorvalue) &
  #   !is.na(colors$colorchroma)
  #
  # colors$color_w <- NA
  # colors$color_d <- NA
  #
  # colors$color_w[idx_w] <- paste(colors$colorhue[idx_w],
  #
paste(colors$colorvalue[idx_w],colors$colorchroma[idx_w],sep='/'),sep=' ')
  # colors$color_d[idx_d] <- paste(colors$colorhue[idx_d],
  #
paste(colors$colorvalue[idx_d],colors$colorchroma[idx_d],sep='/'),sep=' ')
  #
  # colors_w <- colors[!is.na(colors$color_w),]
  # colors_d <- colors[!is.na(colors$color_d),]
  #
  # colors_w2 <- data.frame()
  # for(i in unique(colors_w$labsampnum)) {
  #
  #   a <- dplyr::filter(colors_w,labsampnum==i)[,c('labsampnum','color_w')]
  #
  #   if(nrow(a)==1) {
  #     colors_w2 <- bind_rows(colors_w2,a)
  #     next
```

```
  #   } else {
  #     mixed <- aqp::mixMunsell(a$color_w)
  #     mixed$labsampnum <- i
  #     mixed <- mixed[,c('labsampnum','munsell')]
  #     colnames(mixed) <- c('labsampnum','color_w')
  #
  #     if(is.na(mixed$color_w)) {
  #       mixed$color_w <- a$color_w[1]
  #     }
  #
  #     colors_w2 <- bind_rows(colors_w2,mixed)
  #     next
  #   }
  #
  # }
  #
  # colors_d2 <- data.frame()
  # for(i in unique(colors_d$labsampnum)) {
  #
  #   a <- dplyr::filter(colors_d,labsampnum==i)[,c('labsampnum','color_d')]
  #
  #   if(nrow(a)==1) {
  #     colors_d2 <- bind_rows(colors_d2,a)
  #     next
  #   } else {
  #     mixed <- aqp::mixMunsell(a$color_d)
  #     mixed$labsampnum <- i
  #     mixed <- mixed[,c('labsampnum','munsell')]
  #     colnames(mixed) <- c('labsampnum','color_d')
  #
  #     if(is.na(mixed$color_d)) {
  #       mixed$color_d <- a$color_d[1]
  #     }
  #
  #     colors_d2 <- bind_rows(colors_d2,mixed)
  #     next
  #   }
  #
  # }
  #
  #
  # hzs <-
merge(hzs,colors_w2[,c('labsampnum','color_w')],by='labsampnum',all.x=TRUE)
  # hzs <-
merge(hzs,colors_d2[,c('labsampnum','color_d')],by='labsampnum',all.x=TRUE)




  ##### Structure #####
```

```
  struct$structgrade[struct$structtype=='Massive' | struct$structtype=='Single
grain'] <- 'Structureless'
  struct <- struct[!(!is.na(struct$structgrade) & !is.na(struct$structsize) &
is.na(struct$structtype)),]

  struct <- dplyr::filter(struct,!is.na(structgrade))
  rmidx <- is.na(struct$structsize) & struct$structtype != "Massive"
  struct <- struct[!rmidx,]

  hzs$structgrade <- NA
  hzs$structsize <- NA
  hzs$structtype <- NA
  for(i in c(1:nrow(hzs))) {
    l <- hzs$labsampnum[i]
    idx <- which(struct$labsampnum == l)

    if(length(idx)==0) {
      hzs$structgrade[i] <- NA
      hzs$structsize[i] <- NA
      hzs$structtype[i] <- NA
    }

    if(length(idx)==1) {
      hzs$structgrade[i] <- struct$structgrade[idx]
      hzs$structsize[i] <- struct$structsize[idx]
      hzs$structtype[i] <- struct$structtype[idx]
    }

    if(length(idx)==2) {

      if((!is.na(struct$structgrade[idx[1]]) &
          struct$structgrade[idx[1]] == 'Structureless') |
         (!is.na(struct$structtype[idx[1]]) &
          struct$structtype[idx[1]] == 'Massive')) {
        idx <- idx[1]
        hzs$structgrade[i] <- struct$structgrade[idx]
        hzs$structsize[i] <- struct$structsize[idx]
        hzs$structtype[i] <- struct$structtype[idx]
        next
      }
      if((!is.na(struct$structgrade[idx[1]]) &
          struct$structgrade[idx[2]] == 'Structureless') |
         (!is.na(struct$structtype[idx[2]]) &
          struct$structtype[idx[2]] == 'Massive')) {
        idx <- idx[2]
        hzs$structgrade[i] <- struct$structgrade[idx]
        hzs$structsize[i] <- struct$structsize[idx]
        hzs$structtype[i] <- struct$structtype[idx]
        next
      }

      # change to tolower
      stsize <- c('Very fine','Fine','Medium','Coarse','Very
coarse','Extremely coarse')
```

```
      plstsize <- c('Very thin','Thin','Medium','Thick','Very thick')


      struct$structsize[idx][struct$structsize[idx] == "Fine and medium"] <-
"Fine"
      struct$structsize[idx][struct$structsize[idx] == "Medium and coarse"] <-
"Coarse"
      struct$structsize[idx][struct$structsize[idx] == "Very fine and fine"]
<- "Very fine"


      if(struct$structtype[idx[1]] == 'Platy') {
        i1 <- which(plstsize %in% struct$structsize[idx[1]])
        if(length(i1)==0) {
          i1 <- which(stsize %in% struct$structsize[idx[1]])
        }
      } else {
        i1 <- which(stsize %in% struct$structsize[idx[1]])
      }
      if(struct$structtype[idx[2]] == 'Platy') {
        i2 <- which(plstsize %in% struct$structsize[idx[2]])
        if(length(i2)==0) {
          i2 <- which(stsize %in% struct$structsize[idx[2]])
        }
      } else {
        i2 <- which(stsize %in% struct$structsize[idx[2]])
      }

      if(i1 < i2 | i1 == i2) {
        idx <- idx[1]
        hzs$structgrade[i] <- struct$structgrade[idx]
        hzs$structsize[i] <- struct$structsize[idx]
        hzs$structtype[i] <- struct$structtype[idx]
        next
      }

      if(i2 < i1) {
        idx <- idx[2]
        hzs$structgrade[i] <- struct$structgrade[idx]
        hzs$structsize[i] <- struct$structsize[idx]
        hzs$structtype[i] <- struct$structtype[idx]
        next
      }


    }


  } # end loop



  #### Frags ####
```

```
  frags <- frags[!duplicated(frags),]
  frags <-
soilDB::simplifyFragmentData(rf=frags,id.var='labsampnum',vol.var='fragvol')
  frags$total_frags_pct[frags$total_frags_pct > 100] <- 100 # quick fix

  hzs <-
merge(hzs,frags[,c('labsampnum','total_frags_pct')],by='labsampnum',all.x=TRUE)



  #### Redox ####

  redox <-
redox[,c('labsampnum','rdxfeatkind','colorhue','colorvalue','colorchroma')]
  redox$redoxconcen <- grepl('concentrations|masses|concretions|
nodules',tolower(redox$rdxfeatkind))
  redox$redoxmass <- grepl('concentrations|masses|concretions|
nodules',tolower(redox$rdxfeatkind))
  redox$redoxconcre <- grepl('concretions',tolower(redox$rdxfeatkind))
  redox$redoxdepl <- grepl('depletions|depleted|
reduced',tolower(redox$rdxfeatkind))

  colnames(redox) <-
c('labsampnum','rdxfeatkind','redoxcolorhue','redoxcolorvalue','redoxcolorchroma',
                       'redoxconcen','redoxmass','redoxconcre','redoxdepl')

  hzs <- merge(hzs,redox,by='labsampnum',all.x=TRUE)



  #### Filter again ####

  # bs82 - needed for ultisols/alfisols
  # cec7 - cec7; needed for kandic, oxic
  # claytotal - clay; exclude O, R
  # color_d - color_d
  # color_w - color_w
  # dboevendry - db_od
  # dbthirdbar - db_13b
  # hzdepb - hzn_top
  # hzdept - hzn_bot
  # hzname - hzn_desgn
  # ksat - Ks (logcm/d); only needed for albaqualfs <=1.0
  # pedon_id - pedon_key
  # ph1to1h2o - ph_h2o; only needed for spodic_mat
  # rrc - NA
  # sandtotal - sand; exclude O, R



  ## use this for now until we have estimation model
  idx <- is.na(hzs$color_d) & !is.na(hzs$color_w)
  hzs$color_d[idx] <- hzs$color_w[idx]
```

```
  idx <- !is.na(hzs$color_d) & is.na(hzs$color_w)
  hzs$color_w[idx] <- hzs$color_d[idx]


  ## This won't work without SPC
  # v <- c('hzn_desgn', 'sand', 'clay', 'bs82', 'cec7', 'color_d', 'color_w',
'm_value', 'm_chroma')
  #
  # res$fractionComplete <- evalMissingData(res, vars = v, name = 'hzn_desgn',
p = '[RO]')
  #
  # res <- subset(res, fractionComplete == 1)


  ## get ids of pedons we want to remove
  fidx1 <- unique(hzs$pedon_key[is.na(hzs$clay) & is.na(hzs$sand) &
                                      !grepl('[RO]|Cr',hzs$hzn_desgn)])
  fidx2 <- unique(hzs$pedon_key[is.na(hzs$color_d) & is.na(hzs$color_w) &
                                      !grepl('[RO]|Cr',hzs$hzn_desgn)])
  # fidx3 <- unique(hzs$pedon_key[(is.na(hzs$bs82) | is.na(hzs$cec7) |
  #                                  is.na(hzs$db_od) | is.na(hzs$db_13b)) &
  #                                    !grepl('[RO]|Cr',hzs$hzn_desgn)])
  fidx3 <- unique(hzs$pedon_key[(is.na(hzs$bs82) | is.na(hzs$cec7)) &
                                  !grepl('[RO]|Cr',hzs$hzn_desgn)])
  fidx4 <- unique(hzs$pedon_key[is.na(hzs$hzn_desgn)])

  fidx <- unique(c(fidx1,fidx2,fidx3,fidx4))

  hzs <- dplyr::filter(hzs,!pedon_key %in% fidx)
  sites <- dplyr::filter(sites,pedon_key %in% hzs$pedon_key)




  ##### get RRC from SSURGO #####

  drrc <- c('Loose','Soft','Slightly hard','Moderately hard','Hard','Very
hard',
            'Extremely hard','Rigid','Very rigid')
  mrrc <- c('Loose','Very friable','Friable','Firm','Very firm','Extremely
firm',
            'Slightly rigid','Rigid','Very rigid')

  hzs2 <- data.frame()
  for(i in c(1:nrow(sites))) {

    s <- tolower(sites$taxonname[i])
    pk <- sites$pedon_key[i]
    d <- dplyr::filter(hzs,pedon_key==pk)

    q <- SDA_query(paste0(" SELECT ch.*, chc.*
               FROM

                     component     AS co
```

```
              JOIN  chorizon      AS ch  ON co.cokey = ch.cokey
              JOIN  chconsistence AS chc ON ch.chkey = chc.chkey

              WHERE
                    co.compname = '",s,"'

               "))


  for(j in c(1:nrow(d))) {

    if(is.null(q)) {
      d$rrc[j] <- 'Slightly hard'
      next
    }

    qidx1 <- which(q$hzname == d$hzn_desgn[j])
    if(length(qidx1)==0) {
      qidx2 <- which(q$desgnmaster == d$hzn_desgn[j])
      if(!is.na(q$rupresblkdry[qidx2[1]])){
        d$rrc[j] <- q$rupresblkdry[qidx2[1]]
      } else if(!is.na(q$rupresblkmst[qidx2[1]])) {
        d$rrc[j] <- drrc[which(q$rupresblkmst[qidx2[1]]==mrrc)]
      } else {
        d$rrc[j] <- 'Slightly hard'
      }

    } else {
      if(!is.na(q$rupresblkdry[qidx1[1]])){
        d$rrc[j] <- q$rupresblkdry[qidx1[1]]
      } else if(!is.na(q$rupresblkmst[qidx1[1]])) {
        d$rrc[j] <- drrc[which(q$rupresblkmst[qidx1[1]]==mrrc)]
      } else {
        d$rrc[j] <- 'Slightly hard'
      }
    }

  }

  hzs2 <- bind_rows(hzs2,d)

} # end loop


hzs2 <- hzs2[!duplicated(hzs2$labsampnum),]


## replace missing STRs
idx_na <- which(is.na(sites$taxtempregime))
idx_s <- tolower(sites$taxonname[idx_na])
for(i in c(1:length(idx_na))) {
  dx <- idx_na[i]
  sr <- idx_s[i]
  str <- series_str$str[tolower(series_str$series) == sr]
```

```
    sites$taxtempregime[dx] <- str
  }

  ## add smr
  sites$smr <- NA
  for(i in c(1:nrow(sites))) {
    idx <- tolower(series_str$series) == tolower(sites$taxonname[i])
    sites$smr[i] <- series_str$smr[idx]

  }


  dgs <-
diag_struct(unique(sites$pedon_key),smr=sites$smr,str=sites$taxtempregime)


  vars <- c("pedon_id",
            "hzname",
            "desgnmaster",
            "hzdept",
            "hzdepb",
            "hzthk",
            "aadipyridyl",
            "acec7",
            "aecec",
            "albic_vol",
            "aloxalate",
            "al_kcl",
            "andic",
            "anhyd",
            "anthraquiccond",
            "aquic",
            "art",
            "bioturb_vol",
            "bs7",
            "bs82",
            "caco3",
            "cec7",
            "cec82",
            "cemagt",
            "cemented",
            "cindfragpum",
            "clay_films",
            "claysizedcarb",
            "claytotal",
            "color_d",
            "color_w",
            "colorchroma_d",
            "colorchroma_w",
            "colorhue_d",
            "colorhue_w",
            "colorvalue_d",
            "colorvalue_w",
            "consec_sat",
```

```
"cryoturb",
"cumusat",
"dbovendry",
"dbthirdbar",
"densic",
"ec",
"ec_15",
"ecec",
"effclass",
"esp",
"ex_ca",
"ex_mg",
"ex_na",
"extracid",
"failure",
"feoxalate",
"feoxide", # no longer in use
"fragtotvol",
"gelic_mat",
"gypsum",
"humanTpAlt",
"ksat",
"lamellae_thk",
"lamellae_vol",
"lithdisc",
"lithic",
"melanic_idx",
"midden",
"moss_vol",
"n",
"om",
"oc",
"p_ret",
"paralithic",
"permafrost",
"ph1to1h2o",
"ph_cacl2",
"plinthite_vol",
"redoxcolor",
"redoxcolorchroma",
"redoxcolorhue",
"redoxcolorvalue",
"redoxconcen",
"redoxconcre",
"redoxdepl",
"redoxmass",
"rll",
"rrc",
"sandtotal",
"sar",
"sat_mech",
"seccarb",
"silt",
"siltco",
```

```
            "slake_h2o",
            "slake_hcl",
            "slake_koh",
            "slake_naoh",
            "sphgm_vol",
            "spodic_mat_pct",
            "structgrade",
            "structsize",
            "structtype",
            "sulfidic_mat",
            "texture",
            "texturemod",
            "vglass",
            "whc",
            "wormhole_vol")

  hzs <- data.frame(matrix(nrow=nrow(hzs2),ncol=length(vars)))
  colnames(hzs) <- vars

  hzs$aloxalate <- hzs2$al_ox
  hzs$al_kcl <- hzs2$al_kcl
  hzs$bs7 <- hzs2$bs7
  hzs$bs82 <- hzs2$bs82
  hzs$caco3 <- hzs2$caco3
  hzs$cec7 <- hzs2$cec7
  hzs$cec82 <- hzs2$cec82
  hzs$claysizedcarb <- hzs2$co3_cly
  hzs$claytotal <- hzs2$clay
  hzs$color_d <- hzs2$color_d
  hzs$color_w <- hzs2$color_w
  hzs$dbovendry <- hzs2$db_od
  hzs$dbthirdbar <- hzs2$db_13b
  hzs$ec <- hzs2$ec_12pre * 2
  hzs$ecec <- hzs2$ecec
  hzs$ex_ca <- hzs2$ex_ca
  hzs$ex_mg <- hzs2$ex_mg
  hzs$ex_na <- hzs2$ex_na
  hzs$extracid <- hzs2$acid_tea
  hzs$feoxalate <- hzs2$fe_ox
  hzs$fragtotvol <- hzs2$total_frags_pct
  hzs$gypsum <- hzs2$gypl20
  hzs$hzdepb <- hzs2$hzn_bot
  hzs$hzdept <- hzs2$hzn_top
  hzs$hzname <- hzs2$hzn_desgn
  hzs$ksat <- (10^as.numeric(hzs2$Ks))*0.115741 # convert from log10(cm/d) to
um/s
  hzs$om <- hzs2$estimated_om
  hzs$oc <- hzs2$estimated_oc
  hzs$p_ret <- hzs2$p_nz
  hzs$pedon_id <- hzs2$pedon_key
  hzs$ph1to1h2o <- hzs2$estimated_ph_h2o
  hzs$ph_cacl2 <- hzs2$ph_cacl2
  hzs$redoxcolorchroma <- hzs2$redoxcolorchroma
  hzs$redoxcolorvalue <-hzs2$redoxcolorvalue
```

```
  hzs$redoxcolorhue <- hzs2$redoxcolorhue
  hzs$redoxconcen <- hzs2$redoxconcen
  hzs$redoxconcre <- hzs2$redoxconcre
  hzs$redoxdepl <- hzs2$redoxdepl
  hzs$redoxmass <- hzs2$redoxmass
  hzs$rrc <- hzs2$rrc
  hzs$sandtotal <- hzs2$sand
  hzs$sar <- hzs2$sar
  hzs$silt <- hzs2$silt
  hzs$siltco <- hzs2$silt_c_psa
  hzs$structgrade <- hzs2$structgrade
  hzs$structsize <- hzs2$structsize
  hzs$structtype <- hzs2$structtype
  hzs$whc <- hzs2$whc


  ### explicit for histosols ###
  histosol_ids <- sites$pedon_key[tolower(sites$taxonname) %in% c('rindge')]
  hzs$cumusat[hzs$pedon_id %in% histosol_ids] <- 30



  ### Return origin data ###
  # data_hz <- dplyr::filter(data_hz,pedon_key %in% orig_class$site_pk)
  orig_diag <- dplyr::filter(res$morph$pediagfeatures, pedon_key %in%
hzs$pedon_id)
  # data_color <- dplyr::filter(data_color,labsampnum %in%
data_hz3$labsampnum)
  # data_struct <- dplyr::filter(data_struct,labsampnum %in%
data_hz3$labsampnum)

  objlist <- list(hzs=hzs,diags=dgs,orig_class=sites,orig_diag=orig_diag)

  return(objlist)


}



## KSSL pedons within CA630
# ca630 <- read_excel("C:/Users/Marc/Desktop/NRCS Project/analysis/
CA630_pedons.xlsx")

########## Analysis ###########

options(warn=1)
m <- merge_kssl_series(s)
options(warn=2)

mchk <- check_inputs(m[['hzs']],m[['diags']],TRUE)
# hzs <- mchk[['hzs']]
# diags <- mchk[['diags']]
```

```
mcd <- calc_diagnostics(hzs=m[['hzs']],diags=m[['diags']],ignore.na=TRUE)
# hzs <- mcd[['hzs']]
# diags <- mcd[['diags']]
# tax <- tax_struct(unique(hzs$pedon_id))

mct <-
calc_taxonomy(hzs=m[['hzs']],diags=m[['diags']],ignore.na=TRUE,tax_level='suborders')
# hzs <- mct[['hzs']]
# diags <- mct[['diags']]
# tax <- mct[['tax']]
# t <- lapply(tax,function(x){dplyr::filter(x,pedon==id)})

calc_tx <- mct[["tax"]][["class_tax"]]
colnames(calc_tx) <-
c('pedon','cl_orders','cl_suborders','cl_greatgroups','cl_subgroups','cl_family')

load(file="C:/Users/Marc/Desktop/NRCS Project/soil_taxonomy/R/soiltaxonomy/R/
data/ST_logic.rda")


orig_diag <- m[['orig_diag']]

comp <- merge(m[['orig_class']],calc_tx,by.x='pedon_key',by.y='pedon')
wrong_ord <- comp[which(tolower(comp$taxorder) != tolower(comp$cl_orders)),]
wrong_ord_2 <- wrong_ord[,c("pedon_key","pedon_id","taxonname","taxorder",
                            "cl_orders","estWetColor","estDryColor","taxtempregime","smr")]

correct_ord <- comp[which(tolower(comp$taxorder) == tolower(comp$cl_orders)),]
correct_ord_2 <- correct_ord[,c("pedon_key","pedon_id","taxonname","taxorder",
                                "cl_orders","estWetColor","estDryColor","taxtempregime","s

wrong_subord <- correct_ord[which(tolower(correct_ord$taxsuborder) !=
tolower(correct_ord$cl_suborders)),] %>%
  filter(tolower(taxsuborder) %in% tolower(ST_logic$suborder))

wrong_subord_2 <-
wrong_subord[,c("pedon_key","pedon_id","taxonname","taxorder",
                                  "taxsuborder","cl_orders","cl_suborders",
                                  "estWetColor","estDryColor","taxtempregime","smr")]

correct_subord <- correct_ord[which(tolower(correct_ord$taxsuborder) ==
tolower(correct_ord$cl_suborders) |
                                    !tolower(correct_ord$taxsuborder) %in%
tolower(ST_logic$suborder)),]
correct_subord_2 <-
correct_subord[,c("pedon_key","pedon_id","taxonname","taxorder",
                                      "taxsuborder","cl_orders","cl_suborders",
                                      "estWetColor","estDryColor","taxtempregime","smr")]

missing <- comp[which(is.na(comp$taxorder)),
                c("pedon_key","pedon_id","taxonname","taxorder",
                  "taxsuborder","cl_orders","cl_suborders",
                  "estWetColor","estDryColor","taxtempregime","smr")]
```

```
### Alfisols - Great Groups ###
alf_ids <- correct_subord$pedon_key[correct_subord$cl_orders == "alfisols"]

alf_tax <-
calc_taxonomy(hzs=m[['hzs']],diags=m[['diags']],ignore.na=TRUE,tax_level='great
groups',pedon_ids=alf_ids)

alf_tax_df <- alf_tax[['tax']][['class_tax']]
colnames(alf_tax_df) <-
c('pedon','cl_orders','cl_suborders','cl_greatgroups','cl_subgroups','cl_family')

alf_comp <-  merge(m[['orig_class']],alf_tax_df,by.x='pedon_key',by.y='pedon')
wrong_gg <- alf_comp[which(tolower(alf_comp$taxgrtgroup) !=
tolower(alf_comp$cl_greatgroups)),
                     c("pedon_key","pedon_id","taxonname","taxorder",
                       "taxsuborder","taxgrtgroup","cl_orders","cl_suborders","cl_greatgro
                       "estWetColor","estDryColor","taxtempregime","smr")]

correct_gg <- alf_comp[which(tolower(alf_comp$taxgrtgroup) ==
tolower(alf_comp$cl_greatgroups) |
                                       !tolower(alf_comp$taxsuborder) %in%
tolower(ST_logic$suborder)),
                       c("pedon_key","pedon_id","taxonname","taxorder",
                         "taxsuborder","taxgrtgroup","cl_orders","cl_suborders","cl_greatg
                         "estWetColor","estDryColor","taxtempregime","smr")]

### Reports ###

#report_diagnostics(mct[['diags']],mct[['meta']],unique(mct[['hzs']]
$pedon_id),fn="C:/Users/Marc/Desktop/NRCS Project/testing/
diag_report_CA630_extended.txt",extended=TRUE,append=FALSE)

### report_diagnostics(mct[['diags']],mct[['meta']],unique(mct[['hzs']]
$pedon_id),fn="C:/Users/Marc/Desktop/NRCS Project/testing/
diag_report_main_analysis.txt",extended=FALSE,append=FALSE)

### report_taxonomy(mct[['tax']],unique(mct[['hzs']]$pedon_id),fn="C:/Users/
Marc/Desktop/NRCS Project/testing/
tax_report_main_analysis.txt",extended=FALSE,append=FALSE)

#report_taxonomy(mct[['tax']],unique(mct[['hzs']]$pedon_id),fn="C:/Users/Marc/
Desktop/NRCS Project/testing/
tax_report_CA630_extended.txt",extended=TRUE,append=FALSE)



report_diagnostics(alf_tax[['diags']],alf_tax[['meta']],alf_ids,fn="C:/Users/
Marc/Desktop/NRCS Project/testing/
diag_report_Alfs_extended.txt",extended=TRUE,append=FALSE)


report_taxonomy(alf_tax[['tax']],alf_ids,fn="C:/Users/Marc/Desktop/NRCS
Project/testing/tax_report_Alfs_extended.txt",extended=TRUE,append=FALSE)
```

```
###### Plotting ######
# 1. bar graph of correct and mismatched by order
# 2. bar graph of correct and mismatched by suborder
# 3. bar graph of correct and mismatched by great group



##### Orders Plot #####
t1 <- as.data.frame(table(correct_ord_2$cl_orders))
t2 <- as.data.frame(table(tolower(wrong_ord_2$taxorder)))

a <-
data.frame(order=as.character(t1$Var1),count=as.numeric(t1$Freq),Result='Matching
Order')
b <-
data.frame(order=as.character(t2$Var1),count=as.numeric(t2$Freq),Result='Different
Order')

d <- merge(a,b,all=TRUE)

d <- d %>%
  group_by(order) %>%
  mutate(label_y = cumsum(count))

order_plot <- ggplot(d, aes(x=order, y=count, fill=Result, label=count)) +
  geom_bar(position='stack',stat='identity') +
  scale_fill_manual("legend", values = c("Matching Order" = "dodgerblue2",
"Different Order" = "light blue")) +
  geom_text(size = 4, position = position_stack(vjust = 0.5)) +
  labs(title = "Results of Order Comparisons") +
  xlab('Order') +
  ylab('Count')

# show(order_plot)


##### Suborders Plot #####
t1 <- as.data.frame(table(correct_subord_2$cl_suborders))
t2 <- as.data.frame(table(tolower(wrong_subord_2$taxsuborder)))

a <-
data.frame(suborder=as.character(t1$Var1),count=as.numeric(t1$Freq),Result='Matching
Suborder')
b <-
data.frame(suborder=as.character(t2$Var1),count=as.numeric(t2$Freq),Result='Different
Suborder')

d <- merge(a,b,all=TRUE)

d <- d %>%
  group_by(suborder) %>%
```

```
  mutate(label_y = cumsum(count))

suborder_plot <- ggplot(d, aes(x=suborder, y=count, fill=Result, label=count))
+
  geom_bar(position='stack',stat='identity') +
  scale_fill_manual("legend", values = c("Matching Suborder" = "chartreuse4",
"Different Suborder" = "chartreuse2")) +
  geom_text(size = 4, position = position_stack(vjust = 0.5)) +
  labs(title = "Results of Suborder Comparisons") +
  xlab('Suborder') +
  ylab('Count') +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))

# show(suborder_plot)




##### Alfisols GG Plot #####
t1 <- as.data.frame(table(correct_gg$cl_greatgroups))
t2 <- as.data.frame(table(tolower(wrong_gg$taxgrtgroup)))

a <-
data.frame(great_group=as.character(t1$Var1),count=as.numeric(t1$Freq),Result='Matching
Great Group')
b <-
data.frame(great_group=as.character(t2$Var1),count=as.numeric(t2$Freq),Result='Different
Great Group')

d <- merge(a,b,all=TRUE)

d <- d %>%
  group_by(great_group) %>%
  mutate(label_y = cumsum(count))

greatgroup_plot <- ggplot(d, aes(x=great_group, y=count, fill=Result,
label=count)) +
  geom_bar(position='stack',stat='identity') +
  scale_fill_manual("legend", values = c("Matching Great Group" = "brown4",
"Different Great Group" = "darkorange2")) +
  geom_text(size = 4, position = position_stack(vjust = 0.5)) +
  labs(title = "Results of Great Group Comparisons (Alfisols)") +
  xlab('Great Group') +
  ylab('Count') +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5))

# show(greatgroup_plot)



# show(order_plot)
# show(suborder_plot)
```

```
# show(greatgroup_plot)



######## Mismatching Histogram ##########



mis_codes <- data.frame(
           stringsAsFactors = FALSE,
                                pedon.id = c(7319L,16879L,17845L,
                                             18529L,20514L,24540L,3224L,
                                             42416L,48357L,48363L,48484L,
48529L,
                                             48656L,54931L,55099L,68852L,
                                             7319L,11580L,11639L,14433L,
                                             14436L,1473L,17162L,17165L,
18537L,
                                             1970L,21272L,22888L,23554L,2642L,
                                             2794L,43101L,7084L,7091L,
                                             10155L,11642L,15097L,15111L,
15112L,
                                             15114L,34051L,40884L,45786L,
                                             55455L,6364L,6992L),
                                   series = c("corning","reedsburg",
                                             "argonaut","wynoose","cherokee",
                                             "swanboy","holland","cherokee",
                                             "cherokee","cherokee","cherokee",
                                             "cherokee","marion","lowell",
                                             "lowell","lowell","corning",
                                             "cherokee","mariposa","lethent",
                                             "lethent","zook","wynoose","wynoose",
                                             "wynoose","drummer","cherokee",
                                             "zook","zook","drummer",
                                             "handford","cherokee","tranquilar",
                                             "tranquilar","pentz","hotaw",
                                             "clarion","clarion","clarion","clarion",
                                             "whiterock","zook","lindley",
                                             "otwood","cajon","auberry"),
                              mismatch.level = c("great group",
                                             "great group","great group",
                                             "great group","great
group","great group",
                                             "great group","great group",
                                             "great group","great group",
                                             "great group","great
group","great group",
                                             "great group","great group",
                                             "great group","great
group","suborder",
                                             "suborder","suborder","suborder",
```

```
                                        "suborder","suborder","suborder",
                                        "suborder","suborder",
                                        "suborder","suborder","suborder",
                                        "suborder","suborder","suborder",
                                        "suborder","suborder","order","order",
                                        "order","order","order","order",
                                        "order","order","order","order",
                                        "order","order"),
                  reported.classification = c("haploxeralfs",
                                        "paleudalfs","palexeralfs","albaqualfs",
                                        "albaqualfs","natrustalfs",
                                        "haploxeralfs","albaqualfs",
                                        "albaqualfs","albaqualfs","hapludalfs",
                                        "albaqualfs","hapludalfs",
                                        "hapludalfs","hapludalfs","hapludalfs",
                                        "haploxeralfs","aqulfs","xerults",
                                        "argids","argids","aquolls",
                                        "aqualfs","aqualfs","aqualfs",
                                        "aquolls","aqualfs","aquolls",
                                        "aquolls","aquolls","orthents","aqualfs",
                                        "udolls","albolls","mollisols",
                                        "alfisols","mollisols",
                                        "inceptisols","alfisols","mollisols",
                                        "inceptisols","inceptisols",
                                        "alfisols","alfisols","entisols",
                                        "alfisols"),
                calculated.classification = c("palexeralfs",
                                        "haplustalfs","haploxeralfs",
                                        "endoaqualfs","endoaqualfs","haplustalfs",
                                        "palexeralfs","endoaqualfs",
                                        "endoaqualfs","endoaqualfs",
                                        "paleudalfs","haploaqaulfs","paleudalfs",
                                        "paleudalfs","paleudalfs",
                                        "paleudalfs","palexeralfs","udalfs",
                                        "humults","cambids","cambids",
                                        "udolls","udalfs","udalfs","udalfs",
                                        "udolls","udalfs","udolls",
                                        "udolls","udolls","fluvents","udalfs",
                                        "ustolls","ustolls","inceptisols",
                                        "ultisols","inceptisols",
                                        "mollisols","mollisols","inceptisols",
                                        "entisols","entisols","mollisols",
                                        "ultisols","aridisols",
                                        "mollisols"),
                                  reason = c("incorrect/old
classification","incomplete data",
                                        "incorrect/old classification",
                                        "incorrect/old classification &
incomplete data",
                                        "incorrect/old
classification","incomplete data",
                                        "incorrect/old classification",
                                        "incomplete data","incomplete
data",
```

```
                                          "incomplete data","incomplete
data",
                                          "incomplete data","incomplete
data",
                                          "incomplete data","incomplete
data",
                                          "incomplete data",
                                          "incorrect/old classification",
                                          "missing data / incomplete data",
                                          "incorrect/old classification",
                                          "incorrect/old
classification","incorrect/old classification",
                                          "incorrect/old classification",
                                          "missing data / incomplete data",
                                          "missing data / incomplete data",
                                          "missing data / incomplete data",
                                          "incorrect/old classification",
                                          "missing data / incomplete data",
                                          "incorrect/old classification",
                                          "incorrect/old classification",
                                          "incorrect/old classification",
                                          "incorrect/old
classification","incomplete data",
                                          "incorrect/old classification",
                                          "incorrect/old classification",
                                          "incomplete data",
                                          "incorrect/old
classification","incomplete data",
                                          "incomplete data","incomplete
data",
                                          "incomplete data",
                                          "incorrect/old classification",
                                          "logic/interpretation/
code","incomplete data",
                                          "incorrect/old classification",
                                          "incorrect/old classification",
                                          "incomplete data"),
                                   code = c("IC","ID","IC",
                                          "IC/ID","IC","ID","IC","ID","ID",
                                          "ID","ID","ID","ID","ID","ID",
                                          "ID","IC","MD/ID","IC","IC",
                                          "IC","IC","MD/ID","MD/ID",
                                          "MD/ID","IC","MD/
ID","IC","IC","IC",
                                          "IC","ID","IC","IC","ID","IC",
                                          "ID","ID","ID","ID","IC",
                                          "LIC","ID","IC","IC","ID"),
                                  notes = c(NA,
                                          "missing hz under argillic",
                                          "could be interpretation
limitations",
                                          "ksat missing; doesn't meet
requirements for abrupt textural change",
```

```
available data",
'n' in horizon names",NA,"ksat missing",
missing",
dry colors",
confirm aquic conditions at right depth",
hz",
hz",
based on Keys language",
confirm aquic conditions at right depth",
confirm aquic conditions at right depth",
confirm aquic conditions at right depth",
based on Keys language",
confirm aquic conditions at right depth",
based on Keys language",
based on Keys language",
based on Keys language",NA,
aqualf","ustic SMR",
doesn't support Albolls",
required for cambic hz",
classify",NA,
required depth",
             )

"ksat too high according to
"SAR under natric threshold; no
"ksat missing","ksat missing",
"estimated dry colors","ksat
"estimated dry colors",
"estimated dry colors","estimated
"estimated dry colors",NA,
"no redox in E hz, no way to
"data support humults",
"clay does not indicate argillic
"clay does not indicate argillic
"data does not support aquolls
"no redox in E hz, no way to
"no redox in E hz, no way to
"no redox in E hz, no way to
"data does not support aquolls
"no redox in E or A, no way to
"data does not support aquolls
"data does not support aquolls
"data does not support aquolls
"provided data does not support
"ustic SMR; available data
NA,NA,NA,NA,NA,NA,
"just under the 15cm thickness
"buried soil; not sure how to
"bs82 just below 35 at the
"calcic hz and aridic SMR",NA)

t1 <- as.data.frame(table(mis_codes$code))
```

```
d <- data.frame(code=t1$Var1,count=t1$Freq)

d <- d %>%
  group_by(code) %>%
  mutate(label_y = cumsum(count))

mismatch_plot <- ggplot(d, aes(x=reorder(code,-count), y=count, fill=code,
label=count)) +
  geom_bar(position='stack',stat='identity') +
  geom_text(size = 4, position = position_stack(vjust = 0.5)) +
  labs(title = "Reasons for Mismatches") +
  xlab('Reason Code') +
  ylab('Count')


# show(mismatch_plot)









########## Demonstration #1 #############

demo_1_hz <- data.frame(
  stringsAsFactors = FALSE,
                          Series =
c("Tanana","Tanana","Tanana","Tanana","Tanana",
                                    "Terra Ceia","Terra Ceia","Adams",
                                    "Adams","Adams","Adams","Adams","Adams",
                                    "Adams","Adams","Adams","Adams","Neer",
                                    "Neer","Neer","Neer","Neer","Neer","Neer",
                                    "Molokai","Molokai","Molokai","Molokai",
                                    "Molokai","League","League","League",
                                    "League","League","League","League","League",
                                    "Rock River","Rock River","Rock River",
                                    "Rock River","Rock River","Lloyd","Lloyd",
                                    "Lloyd","Lloyd","Lloyd","Lloyd",
                                    "Keith","Keith","Keith","Keith","Keith",
                                    "Keith","Casabonne","Casabonne","Casabonne",
                                    "Casabonne","Casabonne","Woodward",
                                    "Woodward","Woodward","Woodward","Woodward",
                                    "Woodward","Etsel","Etsel","Etsel"),
                             STR = c("gelic","gelic","gelic","gelic","gelic",
                                    "hyperthermic","hyperthermic","frigid",
```

```
                                    "frigid","frigid","frigid","frigid","frigid",
                                    "frigid","frigid","frigid","frigid",
                                    "mesic","mesic","mesic","mesic","mesic",
                                    "mesic","mesic","isohyperthermic",
                                    "isohyperthermic","isohyperthermic",
                                    "isohyperthermic","isohyperthermic","hyperthermic",
                                    "hyperthermic","hyperthermic","hyperthermic",
                                    "hyperthermic","hyperthermic","hyperthermic",
                                    "hyperthermic","frigid","frigid",
                                    "frigid","frigid","frigid","thermic","thermic",
                                    "thermic","thermic","thermic","thermic",
                                    "mesic","mesic","mesic","mesic",
                                    "mesic","mesic","mesic","mesic","mesic",
                                    "mesic","mesic","thermic","thermic","thermic",
                                    "thermic","thermic","thermic","mesic",
                                    "mesic","mesic"),
                             SMR = c("aquic","aquic","aquic","aquic","aquic",
                                    "aquic","aquic","udic","udic","udic","udic",
                                    "udic","udic","udic","udic","udic",
                                    "udic","xeric","xeric","xeric","xeric",
                                    "xeric","xeric","xeric","aridic","aridic",
                                    "aridic","aridic","aridic","udic",
                                    "udic","udic","udic","udic","udic","udic",
                                    "udic","aridic","aridic","aridic",
                                    "aridic","aridic","udic","udic","udic","udic",
                                    "udic","udic","ustic","ustic","ustic",
                                    "ustic","ustic","ustic","xeric","xeric",
                                    "xeric","xeric","xeric","ustic","ustic",
                                    "ustic","ustic","ustic","ustic",
                                    "xeric","xeric","xeric"),
                               Order =
c("gelisols","gelisols","gelisols","gelisols",
                                    "gelisols","histosols","histosols",
                                    "spodosols","spodosols","spodosols","spodosols",
                                    "spodosols","spodosols","spodosols",
                                    "spodosols","spodosols","spodosols",
                                    "andisols","andisols","andisols","andisols",
                                    "andisols","andisols","andisols","oxisols",
                                    "oxisols","oxisols","oxisols","oxisols",
                                    "vertisols","vertisols","vertisols",
                                    "vertisols","vertisols","vertisols","vertisols",
                                    "vertisols","aridisols","aridisols",
                                    "aridisols","aridisols","aridisols",
                                    "ultisols","ultisols","ultisols","ultisols",
                                    "ultisols","ultisols","mollisols","mollisols",
                                    "mollisols","mollisols","mollisols",
                                    "mollisols","alfisols","alfisols","alfisols",
                                    "alfisols","alfisols","inceptisols",
                                    "inceptisols","inceptisols","inceptisols",
                                    "inceptisols","inceptisols","entisols",
                                    "entisols","entisols"),
                           Suborder = c("turbels","turbels","turbels","turbels",
                                    "turbels","saprists","saprists","orthods",
                                    "orthods","orthods","orthods","orthods",
```

```
                    "orthods","orthods","orthods","orthods",
                    "orthods","xerands","xerands","xerands",
                    "xerands","xerands","xerands","xerands",
                    "torrox","torrox","torrox","torrox",
                    "torrox","uderts","uderts","uderts","uderts",
                    "uderts","uderts","uderts","uderts",
                    "argids","argids","argids","argids","argids",
                    "udults","udults","udults","udults",
                    "udults","udults","ustolls","ustolls",
                    "ustolls","ustolls","ustolls","ustolls",
                    "xeralfs","xeralfs","xeralfs","xeralfs",
                    "xeralfs","ustepts","ustepts","ustepts",
                    "ustepts","ustepts","ustepts","orthents",
                    "orthents","orthents"),
      pedon_id = c("gel_1","gel_1","gel_1","gel_1","gel_1",
                    "hist_1","hist_1","spod_1","spod_1","spod_1",
                    "spod_1","spod_1","spod_1","spod_1",
                    "spod_1","spod_1","spod_1","and_1","and_1",
                    "and_1","and_1","and_1","and_1",
                    "and_1","ox_1","ox_1","ox_1","ox_1","ox_1",
                    "vert_1","vert_1","vert_1","vert_1",
                    "vert_1","vert_1","vert_1","vert_1","arid_1",
                    "arid_1","arid_1","arid_1","arid_1",
                    "ult_1","ult_1","ult_1","ult_1","ult_1",
                    "ult_1","moll_1","moll_1","moll_1",
                    "moll_1","moll_1","moll_1","alf_1","alf_1",
                    "alf_1","alf_1","alf_1","incept_1",
                    "incept_1","incept_1","incept_1","incept_1",
                    "incept_1","ent_1","ent_1","ent_1"),
        hzname = c("Oi",
                    "OA","Bjjg","Cjjg","Cjjgf","Oap","Oa",
                    "Oi","Oa","E","Bh","Bhs","Bs","BC",
                    "C1","C2","C3","Oi","A1","A2","Bw1",
                    "Bw2","2Bw3","3Cr","Ap1","Ap2","Bto1",
                    "Bto2","Bto3","Ap","Bw","Bss1","Bss2",
                    "Bss3","Bssg1","Bssg2","Bssg3","A","Bt1",
                    "Bt2","Btk","Bk","Ap","Bt1","Bt2",
                    "Bt3","BC","C","Ap","A","Bt1","Bt2",
                    "BCk","C","A1","A2","Bt1","Bt2","Bt3","A",
                    "BA","Bw","Bk","C","Cr","A1","A2",
                    "R"),
   desgnmaster = c("O",
                    "OA","B","C","C","O","O","O","O","E",
                    "B","B","B","BC","C","C","C","O",
                    "A","A","B","B","B","C","A","A","B",
                    "B","B","A","B","B","B","B","B","B",
                    "B","A","B","B","B","B","A","B","B",
                    "B","BC","C","A","A","B","B","BC",
                    "C","A","A","B","B","B","A","BA","B",
                    "B","C","C","A","A","R"),
        hzdept = c(0L,
                    8L,14L,44L,63L,0L,20L,0L,1L,3L,10L,
                    17L,34L,43L,61L,88L,145L,0L,5L,10L,
                    18L,28L,45L,71L,0L,18L,38L,89L,163L,
```

```
                    0L,15L,28L,56L,76L,91L,117L,150L,0L,
                    5L,25L,48L,66L,0L,23L,43L,84L,115L,
                    142L,0L,18L,33L,48L,76L,112L,0L,
                    10L,28L,66L,91L,0L,24L,37L,60L,103L,
                    130L,0L,10L,36L),
       hzdepb = c(8L,
                    14L,44L,63L,183L,20L,165L,1L,3L,10L,
                    17L,34L,43L,61L,88L,145L,190L,5L,10L,
                    18L,28L,45L,71L,94L,18L,38L,89L,
                    163L,183L,15L,28L,56L,76L,91L,117L,
                    150L,203L,5L,25L,48L,66L,152L,23L,43L,
                    84L,115L,142L,180L,18L,33L,48L,76L,
                    112L,152L,10L,28L,66L,91L,124L,24L,
                    37L,60L,103L,130L,170L,10L,36L,37L),
        hzthk = c(8L,
                    6L,30L,19L,120L,20L,145L,1L,2L,7L,7L,
                    17L,9L,18L,27L,57L,45L,5L,5L,8L,
                    10L,17L,26L,23L,18L,20L,51L,74L,20L,
                    15L,13L,28L,20L,15L,26L,33L,53L,5L,
                    20L,23L,18L,86L,23L,20L,41L,31L,27L,
                    38L,18L,15L,15L,28L,36L,40L,10L,18L,
                    38L,25L,33L,24L,13L,23L,43L,27L,
                    40L,10L,26L,1L),
  aadipyridyl = c(NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA),
        acec7 = c(NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA),
        aecec = c(NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA),
    albic_vol = c(NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
```

```
                    NA),
          aloxalate = c(NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,1.61,2.16,2.2,2.19,
                    2.61,4.77,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA),
             al_kcl = c(NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA),
              andic = c(NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,TRUE,TRUE,TRUE,TRUE,
                    TRUE,TRUE,TRUE,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA),
              anhyd = c(NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA),
     anthraquiccond = c(NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA),
              aquic = c(FALSE,
                    FALSE,TRUE,TRUE,TRUE,TRUE,TRUE,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,FALSE,
                    FALSE,FALSE,FALSE,FALSE,TRUE,TRUE,
                    TRUE,FALSE,FALSE,FALSE,FALSE,FALSE,FALSE,
                    FALSE,FALSE,FALSE,FALSE,FALSE,FALSE,
                    FALSE,FALSE,FALSE,FALSE,FALSE,FALSE,
                    FALSE,FALSE,FALSE,FALSE,FALSE,FALSE,FALSE),
               art = c(FALSE,
                    FALSE,FALSE,FALSE,FALSE,FALSE,FALSE,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
```

```
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA),
bioturb_vol = c(NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA),
         bs7 = c(47L,
                    50L,100L,100L,100L,98L,100L,37L,28L,
                    13L,3L,0L,1L,1L,6L,1L,1L,NA,39L,
                    22L,22L,14L,10L,4L,95L,95L,91L,95L,
                    100L,97L,100L,100L,100L,100L,100L,86L,
                    100L,100L,100L,100L,100L,100L,80L,51L,
                    36L,63L,76L,45L,100L,97L,95L,100L,
                    100L,100L,22L,20L,27L,44L,53L,100L,
                    100L,100L,100L,100L,100L,87L,70L,NA),
        bs82 = c(18L,
                    34L,76L,84L,NA,75L,74L,NA,NA,8L,1L,
                    0L,0L,0L,3L,1L,1L,NA,43L,19L,20L,
                    14L,10L,5L,55L,55L,55L,60L,81L,87L,
                    81L,84L,93L,78L,77L,82L,82L,94L,95L,
                    95L,95L,95L,47L,37L,24L,35L,42L,28L,
                    79L,85L,91L,100L,100L,100L,19L,17L,
                    23L,38L,47L,93L,100L,100L,100L,100L,
                    100L,NA,NA,NA),
       caco3 = c(0L,
                    0L,0L,0L,0L,0L,0L,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,0L,0L,0L,0L,10L,NA,NA,NA,
                    NA,NA,NA,0L,0L,0L,1L,9L,4L,NA,NA,
                    NA,NA,NA,0L,7L,21L,22L,31L,23L,0L,
                    0L,NA),
        cec7 = c(63.3,
                    49.1,12.8,6.9,6.9,189.7,156,107.5,
                    137.3,6,14.7,8,3.9,3.1,1.7,1.6,1.6,NA,
                    53.6,22.9,15.8,15.8,15.9,31,6,7,6,
                    7,7,45.8,43.6,41.1,41.9,42.1,44.3,
                    50.2,44.5,20.7,20.7,22,24.8,24.1,8.6,
                    8.3,6.9,5.1,3.3,2.9,14.3,16.4,29.9,
                    25.2,20,18.9,25.4,20.3,17,19.2,22.4,
                    13.1,12.1,12,6.2,5.9,5.2,9.2,7.1,NA),
       cec82 = c(161.4,
                    72.1,22.3,NA,NA,248.6,214.6,NA,NA,
                    9.5,33.1,22.9,11,6.7,3.1,3.5,3.5,NA,
                    49.2,26.1,17.7,15.9,15.4,25.1,NA,5,
                    5,5,5,51.3,54.7,51.3,47.1,55.6,59.1,
                    52.8,55.3,NA,NA,NA,NA,NA,14.8,11.3,
```

```
                    10.6,8.8,6,4.6,18.8,18.8,31.4,30,25,
                    23,29.3,23.3,20.1,22,25,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA),
         cemagt = c(NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA),
       cemented = c(FALSE,
                    FALSE,FALSE,FALSE,FALSE,FALSE,FALSE,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA),
    cindfragpum = c(NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA),
     clay_films = c(NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA),
  claysizedcarb = c(NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,0,1.7,2.3,2.3,1.4,1.1,NA,
                    NA,NA),
      claytotal = c(0,0,
                    10.6,5.5,5.5,16.8,17.3,NA,NA,6.2,
                    8.7,2.7,2.7,2.6,1.5,2.1,2.1,NA,5.2,
                    4.8,4.3,4.3,4.5,3.1,42.7,47.9,45.8,47,
                    46,66.2,62.1,60,59.8,63.4,71.5,70.8,
                    71.7,13,26.3,37.3,41.2,38.8,25.4,
                    38.4,40.6,46.9,36.4,24.9,17.4,22.6,39.3,
                    27.7,18.5,9.6,15.6,17.6,23.3,26.4,
                    26.1,16.1,16.5,16.9,10.8,8.6,7.6,17.6,
                    27.8,NA),
        color_d = c(NA,
```

```
                                  NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                                  NA,NA,NA,NA,NA,NA,"10YR 4/3",
                                  "10YR 4/3","10YR 5/4","10YR 6/4","10YR
6/4",NA,
                                  "2.5YR 3/6","2.5YR 3/6","2.5YR 4/6",
                                  "2.5YR 4/6","5YR 3/4","10YR 5/1","10YR
4/1",
                                  "10YR 4/1","10YR 5/1","10YR 5/1","2.5Y
6/1",
                                  NA,NA,"10YR 6/2","10YR 5/4","10YR 5/4",
                                  "10YR 5/4","10YR 6/2","2.5YR 3/3",
                                  "2.5YR 3/6","2.5YR 3/6","2.5YR 4/6",
                                  "2.5YR 4/8","2.5YR 4/8","10YR 4/2","10YR
4/2",
                                  "10YR 4/2","10YR 5/2","10YR 7/2","10YR
7/2",
                                  "7.5YR 4/4","7.5YR 4/4","7.5YR 5/4",
                                  "7.5YR 5/4","7.5YR 6/6","5YR 4/4","5YR
4/4",
                                  "5YR 5/4","5YR 5/4","2.5YR 5/4",
                                  "2.5YR 5/4","10YR 6/4","10YR 6/4",NA),
                       color_w = c("7.5YR 3/2","10YR 2/1","10R 4/5","5R 4/2",
                                  "5R 4/2","N 2/0","10YR 2/2","7.5YR 2.5/2",
                                  "10YR 2/1","5YR 5/2","2.5YR 2.5/2",
                                  "5YR 3/3","7.5YR 4/6","10YR 5/4","10YR
5/3",
                                  "10YR 6/4","10YR 5/3",NA,"N 2/0","10YR
2/2",
                                  "10YR 3/4","10YR 4/4","10YR 4/6",NA,
                                  "2.5YR 3/4","2.5YR 3/4","2.5YR 3/4",
                                  "2.5YR 3/4","5YR 3/3","10YR 4/1","10YR
3/1",
                                  "10YR 3/1","10YR 3/1","10YR 4/1","2.5Y
5/1",
                                  "2.5Y 6/1","2.5Y 6/1","10YR 3/4",
                                  "10YR 4/4","10YR 4/4","10YR 4/4","10YR
5/3",NA,
                                  NA,NA,NA,NA,NA,"10YR 3/2","10YR 3/2",
                                  "10YR 3/2","10YR 4/2","10YR 6/3",
                                  "10YR 6/3","5YR 3/4","5YR 3/4","5YR 4/4",
                                  "5YR 4/4","7.5YR 4/4","5YR 3/4","5YR 4/4",
                                  "5YR 4/4","5YR 4/4","2.5YR 3/6","2.5YR
3/6",
                                  "10YR 4/4","10YR 4/4",NA),
                  colorchroma_d = c(NA,
                                  NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                                  NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                                  NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                                  NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                                  NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                                  NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                                  NA),
                  colorchroma_w = c(NA,
                                  NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
```

```
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA),
  colorhue_d = c(NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA),
  colorhue_w = c(NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA),
colorvalue_d = c(NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA),
colorvalue_w = c(NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA),
  consec_sat = c(NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA),
    cryoturb = c(FALSE,
                    FALSE,TRUE,TRUE,TRUE,FALSE,FALSE,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA),
```

```
   cumusat = c(NA,
               NA,NA,NA,NA,30L,30L,NA,NA,NA,NA,NA,
               NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
               NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
               NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
               NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
               NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
               NA),
 dbovendry = c(0.08,
               0.1,1.53,1.53,1.53,0.53,0.53,0.9,
               1.075,1.35,1.35,1.35,1.35,1.49,1.49,1.49,
               1.49,0.1,0.56,0.56,0.95,0.95,0.95,
               1.52,1.4,1.4,1.43,1.5,1.45,1.8,1.86,
               1.95,1.83,1.93,1.94,1.99,2.01,1.36,
               1.49,1.49,1.41,1.46,1.47,1.47,1.47,1.47,
               1.47,1.42,1.42,1.45,1.69,1.71,1.51,
               1.42,1.32,1.32,1.19,1.14,1.42,1.46,1.4,
               1.29,1.55,1.62,1.78,1.47,1.42,NA),
dbthirdbar = c(0.08,
               0.1,1.5,1.48,1.48,0.37,0.12,0.9,
               1.075,1.35,1.35,1.35,1.35,1.49,1.49,1.49,
               1.49,0.1,0.54,0.54,0.94,0.94,0.94,
               1.49,1.3,1.3,1.34,1.42,1.34,1.2,1.24,
               1.24,1.26,1.19,1.17,1.17,1.21,1.3,1.45,
               1.45,1.4,1.45,1.47,1.47,1.47,1.47,
               1.47,1.42,1.35,1.36,1.3,1.37,1.36,1.35,
               1.32,1.32,1.19,1.14,1.42,1.37,1.31,
               1.22,1.5,1.6,1.76,1.47,1.42,NA),
    densic = c(NA,
               NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
               NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
               NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
               NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
               NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
               NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
               NA),
        ec = c(0,0,
               0,0,0,3.42,4.06,0,0,0,0,0,0,0,0,
               0,0,0,0,0,0,0,0,0,0.71,0.71,
               1.35,1.27,1.54,1.7,1.88,2.12,1.75,1.5,
               1.34,1.38,1.31,0.15,0.17,0.23,0.42,
               0.68,0,0,0,0,0,0,1,1,1,1,1,1,0,0,
               0,0,0,0.48,0.42,0.36,0.26,0.24,0.26,
               0.3,0.3,NA),
     ec_15 = c(0L,
               0L,0L,0L,0L,NA,NA,NA,NA,NA,NA,NA,
               NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
               NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
               NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
               NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
               NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
               NA),
      ecec = c(NA,
               NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
```

```
                                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                                    NA),
                         effclass = c(NA,
                                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                                    NA,NA,NA,"slightly effervescent",
                                    "slightly effervescent","slightly
effervescent",
                                    "strongly effervescent",
                                    "slightly effervescent","slightly
effervescent",NA,NA,NA),
                           esp = c(0L,
                                    0L,0L,0L,0L,3L,3L,0L,0L,0L,0L,0L,
                                    0L,0L,0L,0L,0L,0L,0L,0L,0L,0L,0L,
                                    0L,3L,3L,6L,5L,6L,4L,4L,4L,4L,4L,
                                    5L,4L,5L,1L,1L,6L,20L,20L,8L,8L,3L,
                                    14L,9L,10L,1L,1L,1L,4L,8L,10L,0L,
                                    1L,1L,2L,1L,NA,NA,NA,NA,NA,NA,NA,
                                    NA,NA),
                        ex_ca = c(22.8,
                                    19.8,13,37.2,37.2,12.8,15.9,35.3,
                                    33.9,0.6,0.3,NA,NA,NA,NA,NA,NA,NA,
                                    18.5,4.2,1.7,1.3,1.3,0.9,4.7,4.7,3.8,
                                    4.5,4.7,32.7,32.9,31.5,31.8,30.7,31.5,
                                    29.9,31.3,19.9,19.9,54,53.1,51.1,4.6,
                                    2.7,1.9,1.9,1.3,0.6,10.1,10.2,16.5,
                                    36.4,44.3,35.3,3.7,2.1,2.2,5,6,15,
                                    39.8,43.9,43.7,31,39.7,6.2,3.2,NA),
                        ex_mg = c(3.3,
                                    4.3,3.8,2.2,2.2,8.8,10.1,3.7,3.7,0.1,
                                    NA,NA,NA,NA,NA,NA,NA,NA,2,0.4,
                                    0.2,0.2,0.2,0.2,3.3,3.3,3.7,3.8,4.4,
                                    9.1,8.6,8.6,9.3,9.8,10.8,10.3,10.9,
                                    5.9,5.9,9.1,11.6,11.6,1.6,0.8,0.4,0.6,
                                    0.9,0.4,2.6,4.1,9.4,9.3,8.6,6.7,0.8,
                                    0.6,1.2,2.3,4.8,1.3,1.3,1.6,1.4,
                                    1.7,2.4,1.4,1.4,NA),
                        ex_na = c(0,0,
                                    0,0,0,16.2,19.5,NA,NA,NA,NA,NA,NA,
                                    NA,NA,NA,NA,NA,0,0,0,0,0,0.1,
                                    0.5,0.5,0.8,1,1,2.4,2.6,2.6,2.5,2.4,
                                    2.8,2.7,3.1,0.2,0.2,1.4,4.9,4.8,0.7,
                                    0.7,0.2,0.7,0.3,0.3,0.1,0.1,0.4,0.9,
                                    1.5,1.9,0.1,0.2,0.2,0.3,0.3,0.1,
                                    0.1,0.1,0,0.1,0,0,0,NA),
                      extracid = c(131.9,
                                    47.6,5.4,1.8,1.8,62,54.8,149.5,
```

```
                201.7,8.7,32.7,22.9,11,6.7,3,3.5,3.5,NA,
                28.1,21.1,13.7,13.8,13.8,23.9,7.2,
                7.2,6.7,5.6,6.2,6.7,10.3,8.4,3.3,12.5,
                13.7,9.6,9.7,1.6,NA,NA,NA,NA,7.9,
                7.1,8.1,5.6,3.5,3.3,3.9,2.9,2.9,NA,
                NA,NA,23.6,1.3,15.5,13.6,13.2,1.3,NA,
                NA,NA,NA,NA,NA,NA,NA),
   failure = c(NA,
                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                NA),
 feoxalate = c(NA,
                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                NA,NA,NA,NA,NA,NA,0.25,0.28,0.3,0.24,
                0.28,0.36,NA,NA,NA,NA,NA,NA,NA,NA,
                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                NA,NA,NA),
   feoxide = c(NA,
                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                NA),
fragtotvol = c(0L,
                0L,2L,1L,1L,0L,0L,NA,NA,NA,NA,NA,
                NA,NA,NA,NA,NA,0L,26L,31L,26L,24L,
                32L,53L,0L,0L,0L,0L,0L,0L,0L,0L,0L,
                0L,0L,0L,0L,0L,0L,0L,10L,15L,0L,
                0L,0L,0L,0L,0L,0L,0L,0L,0L,0L,0L,
                20L,25L,10L,10L,20L,0L,0L,0L,4L,3L,
                0L,30L,45L,NA),
 gelic_mat = c(NA,
                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                NA),
    gypsum = c(NA,
                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
```

```
                  NA),
  humanTpAlt = c(NA,
                  NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                  NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                  NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                  NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                  NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                  NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                  NA),
         ksat = c(91,
                  9.17,9.17,9.17,0,92,92,50,20,100,100,
                  100,100,141,141,141,141,91.74,91.74,
                  91.74,91.74,91.74,91.74,0.21,9,9,
                  2.5,2.5,2.5,0.21,0.21,0.21,0.21,0.21,
                  0.21,0.21,0.21,28.23,9.17,9.17,28.23,
                  28.23,9,9,9,9,9,9,6,3,6,6,6,6,
                  9.17,9.17,9.17,9.17,9.17,9,9,9,9,2.7,
                  2.7,10,10,1),
lamellae_thk = c(NA,
                  NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                  NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                  NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                  NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                  NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                  NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                  NA),
lamellae_vol = c(NA,
                  NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                  NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                  NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                  NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                  NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                  NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                  NA),
    lithdisc = c(NA,
                  NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                  NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                  NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                  NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                  NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                  NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                  NA),
      lithic = c(NA,
                  NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                  NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                  NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                  NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                  NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                  NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                  NA),
 melanic_idx = c(NA,
                  NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                  NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                  NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
```

```
                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                NA),
  midden = c(NA,
                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                NA),
moss_vol = c(NA,
                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                NA),
     n = c(NA,
                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                NA),
    om = c(90,
                22,1.55,1.03,0.517,75,75,90,82,0,0,
                0,0,0,0,0,0,50,23.72,6.77,2.86,
                1.86,1.38,0.26,1.75862069,1.75862069,
                0.8620689655,0.3965517241,0.5172413793,4.25828,
                3.1894,2.91356,2.10328,1.06888,0.431,
                0.36204,0.1724,1.5,0.75,0.75,0.75,0.25,
                2.87908,0.91372,0.79304,0.29308,
                0.20688,0.18964,2.36188,1.31024,1.1206,0.7758,
                0.36204,0.12068,7.27528,4.06864,
                1.25852,0.6896,0.53444,2.22396,1.79296,
                1.25852,0.431,0.24136,0.0862,6.24088,2.12052,
                NA),
    oc = c(52.2,
                12.7,0.9,0.6,0.3,43.5,43.5,52.08,
                47.72,0.64,2.16,1.15,0.6,0.41,0.16,0.16,
                0.16,29,13.76,4.0,1.66,1.08,0.8,
                0.15,1.02,1.02,0.5,0.23,0.3,2.47,1.85,
                1.69,1.22,0.62,0.25,0.21,0.1,0.87,
                0.435,0.435,0.435,0.145,1.6698664,0.5299576,
                0.4599632,0.1699864,0.1199904,0.1099912,
                1.37,0.76,0.65,0.45,0.21,0.07,4.22,
                2.36,0.73,0.4,0.31,1.29,1.04,0.73,
                0.25,0.14,0.05,3.62,1.23,NA),
 p_ret = c(0L,
                0L,16L,15L,15L,0L,0L,0L,0L,0L,0L,0L,
```

```
                                0L,0L,0L,0L,0L,50L,62L,73L,77L,
                                77L,81L,97L,43L,43L,NA,NA,NA,NA,NA,
                                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                                NA,NA,NA,NA),
                   paralithic = c(NA,
                                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                                NA),
                   permafrost = c(FALSE,
                                FALSE,FALSE,FALSE,TRUE,FALSE,FALSE,
                                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                                NA,NA,NA,NA,NA,NA),
                    ph1to1h2o = c(4.7,
                                5.6,6.5,6.5,6.7,6.7,6.6,4.2,4.2,4,
                                4.2,5,5.2,5.2,5.2,5.2,5.4,NA,6,6,6,
                                6,6,NA,4.4,4.6,6.5,6.6,6.6,5.9,6.6,
                                6.5,6.7,6.5,6.6,6.9,6.9,7.2,7.4,
                                7.6,8.4,8.4,6.2,6.1,5.5,5.4,5.6,5.4,
                                6.3,6.6,7.1,8.4,8.8,8.9,5.4,5.4,5.2,
                                5.2,5.2,7.7,8.2,8.3,8.5,8.5,8.6,6.3,
                                6,NA),
                     ph_cacl2 = c(NA,
                                NA,NA,NA,NA,6.6,6.5,3.6,3.1,3.2,4,
                                4.5,4.7,4.8,5.2,5,5,NA,NA,NA,NA,NA,
                                NA,NA,NA,NA,NA,NA,NA,5.6,5.9,6.1,
                                6.1,6.3,6.4,6.2,6.4,7.2,7.4,8,8.2,
                                5.5,5.8,5.7,5.2,5,5.2,4.8,5.7,5.8,
                                6.3,7.8,8,8.1,5.3,5.3,4.8,4.8,4.8,7.1,
                                7.7,7.8,7.8,7.9,8,6.3,6,NA),
                 plinthite_vol = c(NA,
                                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                                NA),
                   redoxcolor = c(NA,
                                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                                NA,NA,NA,NA,NA,NA,"7.5YR 5/6",
                                "7.5YR 5/6","7.5YR 5/6","10YR 5/4","2.5Y
5/6",
                                "2.5YR 4/1","10YR 6/6","10YR 5/6",NA,NA,
```

```
                        NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                        NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                        NA,NA,NA,NA,NA,NA,NA),
redoxcolorchroma = c(NA,
                        NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                        NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                        NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                        NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                        NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                        NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                        NA),
   redoxcolorhue = c(NA,
                        NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                        NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                        NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                        NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                        NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                        NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                        NA),
 redoxcolorvalue = c(NA,
                        NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                        NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                        NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                        NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                        NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                        NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                        NA),
     redoxconcen = c(NA,
                        NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                        NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                        NA,NA,NA,NA,NA,NA,TRUE,TRUE,TRUE,
                        TRUE,TRUE,TRUE,TRUE,TRUE,NA,NA,NA,NA,
                        NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                        NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                        NA,NA,NA,NA,NA),
     redoxconcre = c(NA,
                        NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                        NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                        NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                        NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                        NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                        NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                        NA),
       redoxdepl = c(NA,
                        NA,TRUE,TRUE,TRUE,NA,NA,NA,NA,NA,NA,NA,
                        NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                        NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                        TRUE,TRUE,TRUE,NA,NA,NA,NA,NA,NA,NA,
                        NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                        NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                        NA,NA),
       redoxmass = c(NA,
                        NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                        NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
```

```
                                    NA,NA,NA,NA,NA,NA,TRUE,TRUE,TRUE,
                                    TRUE,TRUE,TRUE,TRUE,TRUE,NA,NA,NA,NA,
                                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                                    NA,NA,NA,NA,NA),
                            rll = c(NA,
                                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                                    NA),
                            rrc = c("very friable","very friable","very
friable",
                                    "very friable","extremely firm",
                                    "very friable","very friable","very
friable",
                                    "very friable","very friable","friable",
                                    "very friable","very friable","very
friable",
                                    "loose","loose","loose","very friable",
                                    "very friable","very friable","very
friable",
                                    "very friable","very friable",
                                    "very
friable","friable","friable","friable","firm",
                                    "firiable","extremely firm",
                                    "extremely firm","extremely
firm","extremely firm",
                                    "extremely firm","extremely firm",
                                    "extremely firm","extremely firm","very
friable",
                                    "firm","firm","friable","very friable",
                                    "very
friable","firm","firm","firm","friable",
                                    "friable","friable","friable","friable",
                                    "friable","very friable","very friable",
                                    "very friable","very friable","friable",
                                    "firm","firm","friable","friable",
                                    "friable","friable","friable","friable",
                                    "friable","friable","very friable"),
                        sandtotal = c(0,0,
                                    6.2,7.6,7.6,10.9,28.5,NA,NA,65.2,
                                    78.4,87.1,87.4,87.6,91.8,91.8,85.2,NA,
                                    63.6,68.2,71.2,72.5,71.2,78.4,17,17,
                                    24.1,18.8,12.6,4.1,4.9,4.9,4.4,5,3.7,
                                    3.9,3.3,67.4,52.8,35.8,26.6,26.9,
                                    52.1,39,37.6,33.5,38.3,48.8,38.4,30.4,
                                    19.3,22.7,30.9,54.8,46.8,44.4,37.9,
                                    37.8,43.1,46.4,47.4,46.6,62,69.3,67.6,
                                    55.9,44.3,NA),
                            sar = c(0L,
                                    0L,0L,0L,0L,5L,5L,0L,0L,0L,0L,0L,
```

```
                0L,0L,0L,0L,0L,NA,NA,NA,NA,NA,NA,
                NA,3L,3L,5L,5L,5L,4L,4L,4L,4L,4L,
                4L,4L,5L,0L,0L,0L,0L,2L,0L,0L,0L,
                0L,0L,0L,0L,0L,0L,0L,0L,0L,0L,0L,
                0L,0L,0L,0L,0L,0L,0L,0L,0L,0L,0L,
                NA),
 sat_mech = c(NA,
                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                NA),
  seccarb = c(NA,
                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                NA),
     silt = c(NA,
                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                NA),
   siltco = c(NA,
                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                NA),
slake_h2o = c(NA,
                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                NA),
slake_hcl = c(NA,
                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                NA),
```

```
      slake_koh = c(NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA),
     slake_naoh = c(NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA),
      sphgm_vol = c(NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA),
  spodic_mat_pct = c(NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,100L,
                    100L,100L,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                    NA,NA),
     structgrade = c(NA,
                    "weak","weak","weak","structureless",
                    "weak","weak","structureless","weak","weak",
                    "weak","weak","weak","weak",NA,NA,NA,
                    NA,"moderate","moderate","weak","weak",
                    "weak",NA,"weak","weak","weak","weak",
                    "moderate","weak","weak","weak","weak",
                    "weak","moderate","moderate","moderate",
                    "weak","weak","moderate","weak",
                    "structureless","moderate","weak","moderate",
                    "moderate","weak","structureless",
                    "moderate","moderate","moderate","moderate",
                    "weak","structureless","moderate","moderate",
                    "weak","weak","moderate","moderate",
                    "moderate","moderate","weak","structureless",
                    "structureless","moderate","moderate",NA),
      structsize = c(NA,
                    "fine","medium","medium",NA,"fine",
                    "medium",NA,"fine","fine","medium","medium",
                    "medium","medium",NA,NA,NA,NA,"fine",
                    "fine","medium","medium","medium",NA,
                    "very fine","medium","coarse","coarse",
```

```
                                    "very fine","medium","medium","coarse",
                                    "coarse","coarse","coarse","coarse",
                                    "coarse","fine","medium","medium","medium",NA,
                                    "fine","medium","medium","medium",
                                    "medium",NA,"fine","fine","medium","medium",
                                    "coarse",NA,"fine","fine","medium",
                                    "fine","medium","medium","medium","medium",
                                    "medium",NA,NA,"fine","fine",NA),
                     structtype = c(NA,
                                    "granular","platy","platy","massive",
                                    "granular","subangular blocky","massive",
                                    "granular","subangular blocky",
                                    "subangular blocky","subangular blocky",
                                    "subangular blocky","subangular
blocky","single grain",
                                    "single grain","single
grain",NA,"granular",
                                    "granular","subangular blocky",
                                    "subangular blocky","subangular
blocky",NA,
                                    "granular","subangular
blocky","prismatic",
                                    "prismatic","subangular blocky",
                                    "subangular blocky","subangular
blocky","wedge","wedge",
                                    "wedge","wedge","angular blocky",
                                    "angular blocky","granular","prismatic",
                                    "prismatic","angular
blocky","massive","granular",
                                    "subangular blocky","subangular blocky",
                                    "subangular blocky","subangular blocky",
                                    "massive","granular","granular","prismatic",
                                    "prismatic","subangular blocky","massive",
                                    "subangular blocky","subangular blocky",
                                    "subangular blocky","subangular blocky",
                                    "angular blocky","granular","granular",
                                    "granular","granular","massive","massive",
                                    "subangular blocky","subangular
blocky",NA),
                   sulfidic_mat = c(NA,
                                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                                    NA),
                       texture = c(NA,
                                    "sily loam","silt loam",
                                    "very fine sandy loam","silt
loam",NA,NA,NA,NA,
                                    "loamy fine sand","loamy fine sand",
                                    "loamy fine sand","loamy fine
sand","sand","sand","sand",
```

```
                                    "sand",NA,"sandy loam","sandy loam",
                                    "sandy loam","sandy loam","sandy loam",NA,
                                    "silty clay loam","silty clay loam",
                                    "silty clay loam","silty clay loam","clay
loam",
                                    "clay","clay","clay","clay","clay",
                                    "clay","clay","clay","sandy loam",
                                    "sandy clay loam","sandy clay loam","sandy
loam",
                                    "sandy loam","sandy clay loam","clay
loam",
                                    "clay","clay","clay loam",
                                    "sandy clay loam","loam","loam","silty
clay loam",
                                    "clay loam","silt loam","very fine sandy
loam",
                                    "loam","loam","loam","loam","loam",
                                    "loam","loam","loam","very fine sandy
loam",
                                    "fine sandy loam","fine sandy loam",
                                    "loam","loam",NA),
                     texturemod = c(NA,
                                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                                    NA,NA,NA,NA,NA,NA,"gravelly",
                                    "gravelly","gravelly","very gravelly",
                                    "very
gravelly",NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                                    "gravelly","gravelly",NA,NA,"gravelly",NA,
                                    NA,NA,NA,NA,NA,"gravelly",
                                    "very gravelly",NA),
                         vglass = c(0L,
                                    0L,0L,0L,0L,0L,0L,NA,NA,NA,NA,NA,
                                    NA,NA,NA,NA,NA,NA,43L,40L,36L,31L,
                                    37L,15L,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                                    NA,NA),
                            whc = c(NA,
                                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                                    NA),
                    wormhole_vol = c(NA,
                                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
```

```
                                    NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                                    NA)
             )

demo_1_diags <-
diag_struct(c('gel_1','hist_1','spod_1','and_1','ox_1','vert_1','arid_1',
                               'ult_1','moll_1','alf_1','incept_1','ent_1'),
                             c('aquic','aquic','udic','xeric','aridic','udic','aridic',
                               'udic','ustic','xeric','ustic','xeric'),
                             c('gelic','hyperthermic','frigid','mesic','isohyperthermic',
                               'hyperthermic','frigid','thermic','mesic','mesic','thermic',

demo_1_inputs <- check_inputs(demo_1_hz,demo_1_diags,ignore.na=TRUE)
# hzs <- demo_1_inputs[['hzs']]
# diags <- demo_1_inputs[['diags']]

demo_1_tax <-
calc_taxonomy(demo_1_hz,demo_1_diags,ignore.na=TRUE,tax_level='suborders')
# hzs <- demo_1_tax[['hzs']]
# diags <- demo_1_tax[['diags']]

demo_1_tax_gg <- calc_taxonomy(demo_1_hz,demo_1_diags,ignore.na=TRUE,
                                pedon_ids=c('gel_1','hist_1','alf_1'),tax_level='great
groups')
```

Appendix B:

calc_diagnostics.R

```
## This script contains the main logic for running diagnostic functions.
##
## Main functions contain 2 modules:
## 1. Main pipe for running all diagnostics with correct logic
## 2. Module for running individual functions by name.
##
##
##


# TODO:
# - Review and revise all things related to spodic hz and spodic_mat
calc_diagnostics <- function(hzs, diags='NA', pedon_ids=NA, smr=NA, str=NA,
                             mast=NA, msst=NA, mwst=NA, diagnostics='all',
                             ignore.na=FALSE, override.andic=TRUE) {
  #' @param hzs dataframe with all pedons and horizon data
  #' @param diags diagnostics data structure; inits a new one if not provided
  #' @param pedon_id which pedon_id to apply functions; runs for all if not
specified.
  #' @param smr vector of soil moisture regimes; used to initialize
diagnostics structure
  #' @param str vector of soil temperature regimes; used to initialize
diagnostics structure
  #' @param mast mean annual soil temperature; used to calculate str
  #' @param msst mean summer soil temperature; used to calculate str
  #' @param mwst mean winter soil temperature; used to calculate str
  #' @param diagnostic vector with diagnostic function names to run
  #' @param ignore.na flag for ignoring missing inputs in check inputs
  #' @param override.andic flag for choosing to override input andic property
flag
  #'
  #' @return list of hzs and diags; need to update hzs for rll




  ########## Check for input diag structure ##########

  if(class(diags)=="list") {

    samp_diag <- diag_struct(1)
    samp_names <- unlist(lapply(samp_diag,function(x){colnames(x)}))

    in_names <- unlist(lapply(diags,function(x){colnames(x)}))

    if(!identical(samp_names,in_names)) {

      stop("InputError: Input diag structure is not valid.")

    }


  } else {
```

```
  ## lengths of inputs must be the same OR a single value

  if(all(is.na(pedon_ids))) {

    pedon_ids <- unique(hzs$pedon_id)

  }

  if(!any(pedon_ids %in% unique(hzs$pedon_id))) {

    stop("InputError: no matching pedon_ids in hzs and diag.")

  }

  diags <- diag_struct(pedon_ids,smr,str,mast,msst,mwst)

}




########## Main Logic ##########


## Check inputs ##
checked_inputs <- check_inputs(hzs, diags, ignore.na)
hzs <- checked_inputs[['hzs']]
diags <- checked_inputs[['diags']]


if(all(is.na(pedon_ids))) {

  pedon_ids <- unique(hzs$pedon_id)

}

ids <- pedon_ids



if('all' %in% diagnostics) {


  ### Main Pipe Internal Function to Apply ###
  # takes one id at a time
  # updates hzs and diags
  diag_pipe <- function(id, hzs, diags) {

    print(paste("Calcualting diagnostics for: ",id,sep=''))

    ### Diags Part 1 ###
```

```
      res1 <- duripan(hzs,diags,id) %>%
        fragipan(hzs,.,id) %>%
        petrocalcic(hzs,.,id) %>%
        petrogypsic(hzs,.,id) %>%
        densic(hzs,.,id) %>%
        lithic(hzs,.,id) %>%
        paralithic(hzs,.,id) %>%
        petroferric(hzs,.,id)


        res2 <- albic_mat(hzs,res1,id)
        hzs <- res2[['hzs']]
        diags <- res2[['diags']]


        res3 <- albic(hzs,diags,id) %>%
        placic(hzs,.,id)




      ### Update RLL ###
      rll_tops <-
diag_top_depth(c('duripan','fragipan','petrocalcic','petrogypsic',
                                  'densic','lithic','paralithic','petroferric',
                                  'placic'),res3,id)

      rll_bots <-
diag_bot_depth(c('duripan','fragipan','petrocalcic','petrogypsic',
                                  'densic','lithic','paralithic','petroferric',
                                  'placic'),res3,id)

      idx <- which(hzs$pedon_id==id)

      if(all(is.na(rll_tops))) {

        hzs$rll[idx] <- FALSE

      } else {


        rll_hz <- vector()
        for(i in c(1:length(rll_tops))) {

          rt <- rll_tops[i]
          rb <- rll_bots[i]

          rl <- (hzs$hzdept[idx] >= rt & hzs$hzdepb[idx] <= rb)

          rll_hz <- append(rll_hz,which(rl))


          if(length(rl)==0) {
```

```
      stop("Error: RLL diagnostics don't line up with horizon bounds.")
    }
    hzs$rll[idx][rll_hz] <- TRUE
  }
}



### Diags Part 2 ###
res4 <- orgmin(hzs,res3,id) %>%
  agric(hzs,.,id) %>%
  argillic(hzs,.,id) %>%
  calcic(hzs,.,id) %>%
  glossic(hzs,.,id) %>%
  gypsic(hzs,.,id) %>%
  kandic(hzs,.,id) %>%
  natric(hzs,.,id) %>%
  ortstein(hzs,.,id)

if(override.andic) {
  res5 <- andic_properties(hzs,res4,id)
  hzs <- res5[['hzs']]
  diags <- res5[['diags']]
} else{
  hzs <- res4[['hzs']]
  diags <- res4[['diags']]
}


res6 <- oxic(hzs,diags,id) %>%
  salic(hzs,.,id) %>%
  spodic(hzs,.,id) %>%
  epi.anthropic(hzs,.,id) %>%
  epi.folistic(hzs,.,id) %>%
  epi.histic(hzs,.,id) %>%
  epi.melanic(hzs,.,id) %>%
  epi.mollic(hzs,.,id) %>%
  epi.umbric(hzs,.,id) %>%
  epi.ochric(hzs,.,id) %>%
  epi.epi(.,id) %>%
  spodic_mat(hzs,.,id) %>%
  abrupt_text(hzs,.,id) %>%
  cambic(hzs,.,id)
## run partctrl after orders


retlist <- list(hzs=hzs,diags=res6)
return(retlist)
```

```
    }


    for(id in ids) {

      pipe_res <- diag_pipe(id, hzs, diags)
      hzs <- pipe_res[['hzs']]
      diags <- pipe_res[['diags']]

    }


  } else {

    ## RLL NOT UPDATED IN THIS PATH ##
    warning("not running all diagnostics in order may give inaccurate
results.")
    warning("rll from diagnostics is not updated when calling individual
functions.")

    valid_funs <- c("duripan",
                    "fragipan",
                    "petrocalcic",
                    "petrogypsic",
                    "densic",
                    "lithic",
                    "paralithic",
                    "petroferric",
                    "albic",
                    "spodic_mat",
                    "placic",
                    "agric",
                    "argillic",
                    "calcic",
                    "glossic",
                    "gypsic",
                    "kandic",
                    "natric",
                    "ortstein",
                    "oxic",
                    "salic",
                    "spodic",
                    "abrupt_text",
                    "andic_properties",
                    "albic_mat",
                    "orgmin",
                    "epi.anthropic",
                    "epi.folistic",
                    "epi.histic",
                    "epi.melanic",
                    "epi.mollic",
                    "epi.umbric",
```

```
                    "epi.ochric",
                    "partctrl",
                    "cambic")

    if(any(!diagnostics %in% valid_funs)) {

      w <- diagnostics[which(!diagnostics %in% valid_funs)]

      stop(paste("InputError: not valid diagnostic functions:
",paste(w,collapse=', '),sep=""))

    }



    d <- lapply(ids, function(x) {

        l <- lapply(diagnostics, function(y) {

          f <- match.fun(y)
          res <- f(hzs,diags,x)

          assign('diags',res, envir=parent.frame(3))


      })


    })



  }

  retlist <- list(hzs=hzs,diags=diags)
  return(retlist)


}




################################################################################
## Function to create report of diagnostics ##
# TODO:
# - consider merging dfs for extended report
# - add definitions to end of report
# - gets IDs from inputs if not given
report_diagnostics <- function(diags, meta, ids='NA', fn=NA, extended=FALSE,
append=FALSE) {
  #' @param diags diagnostics data structure
  #' @param meta data structure with input assumptions from check_inputs
  #' @param ids list of ids
```

```
#' @param fn file name or path to text file
#' @param extended flag for writing summary report or full report
#' @param append flag for appending report to existing file


##### Diagnostics Names #####

# epis <- c('anthropic',
#           'folistic',
#           'histic',
#           'melanic',
#           'mollic',
#           'umbric')


diag_hz <- c('agric',
             'albic',
             'argillic',
             'calcic',
             'cambic',
             'densic',
             'duripan',
             'fragipan',
             'glossic',
             'gypsic',
             'kandic',
             'lithic',
             'natric',
             'ortstein',
             'oxic',
             'paralithic',
             'petrocalcic',
             'petroferric',
             'petrogypsic',
             'placic',
             'salic',
             'spodic')


diag_props <- c('abrupt_text',
                'andic_prop',
                'albic_mat',
                'anhydrous_cond',
                'fragic_prop',
                'spodic_mat')


## Delete old file ##
if(!append & file.exists(fn)) {
  file.remove(fn)
}
```

```
  ##### Internal Report Function for ID list #####
  report_l <- function(diags, meta, id, fn, extended) {

    diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

    met <- meta[[as.character(id)]]


    ########## Basic Summary ##########
    if(!extended) {

      #### Get Diagnostic Results ####
      epi <- diag$epipedon
      epi <- epi[,c(5,2,3,4)]
      colnames(epi)[1] <- 'Epipedon'

      true_hz <- diag[diag_hz][as.logical(
        lapply(diag[diag_hz],function(x){x[['hz']]})
        )]

      true_prop <- diag[diag_props][as.logical(
        lapply(diag[diag_props],function(x){x[['prop']]})
      )]

      str <- diag$str$str
      smr <- diag$smr$smr

      orgmin <- diag$orgmin
      colnames(orgmin)[4:7] <- c('organic_top',
'organic_bottom','mineral_top', 'mineral_bottom')


      org_thk <- diag$org_thk
      colnames(org_thk)[6:7] <- c('organic_ctrl_sect_top',
'organic_ctrl_sect_bottom')

      partctrl <- diag$partctrl



      ##### Write Report ####

      if(!is.na(fn)) {
        sink(file=fn,append=TRUE)
      }

      cat(paste(c('----------- Extended Diagnostic Report for Pedon
ID:',id,'-----------\n\n\n'),collapse=' '))
      cat(paste('Soil Temperature Regime:    ',str,sep=' '))
      cat('\n')
      cat(paste('Soil Moisture Regime:       ',smr,sep=' '))
      cat('\n\n\n')
      cat('------------------- Epipedon -------------------\n')
      print(epi)
```

```
  cat('\n\n')
  cat('---------- Subsurface Diagnostic Hzs. ----------\n')

  if(length(true_hz)!=0) {
    print(dplyr::bind_rows(true_hz, .id='Diagnostic_Hz')[,c(1,3,4,5)])
  } else {
    cat('None Detected.')
  }

  cat('\n\n\n')
  cat('---------- Other Diagnostic Features -----------\n')

  if(length(true_prop)!=0) {
    print(dplyr::bind_rows(true_prop, .id='Diagnostic_Feat')[,c(1,3,4,5)])
  } else {
    cat('No diagnostic properties or materials detected.')
  }

  cat('\n\n')
  print(orgmin[,4:7])
  cat('\n')
  print(org_thk[,2:7])
  cat('\n')
  print(partctrl[2:3])
  cat('\n\n\n\n\n')


  if(!is.na(fn)) {
    sink()
  }

}




########## Extended Summary - Raw output##########
if(extended) {

  str <- diag$str$str
  smr <- diag$smr$smr

  epi <- diag$epipedon

  orgmin <- diag$orgmin
  org_thk <- diag$org_thk
  partctrl <- diag$partctrl

  width <- 350

  if(!is.na(fn)) {
    sink(file=fn,append=TRUE)
  }
```

```
      cat(paste(c('--------------- Diagnostic Report for Pedon
ID:',id,'---------------\n\n\n'),collapse=' '))
      cat(paste('Soil Temperature Regime:    ',str,sep=' '))
      cat('\n')
      cat(paste('Soil Moisture Regime:       ',smr,sep=' '))
      cat('\n\n\n')
      cat('------------------- Epipedon -------------------\n')
      print(epi)
      cat('\n\n')
      cat('anthropic','\n')
      print(diag$anthropic,width=width)
      cat('\n')
      cat('folistic','\n')
      print(diag$folistic,width=width)
      cat('\n')
      cat('histic','\n')
      print(diag$histic,width=width)
      cat('\n')
      cat('melanic','\n')
      print(diag$melanic,width=width)
      cat('\n')
      cat('mollic','\n')
      print(diag$mollic,width=width)
      cat('\n')
      cat('umbric','\n')
      print(diag$umbric,width=width)
      cat('\n\n\n')
      cat('---------- Subsurface Diagnostic Hzs. ----------\n')

      for(i in diag_hz) {
        cat(i,'\n')
        print(diag[[i]],width=width)
        cat('\n')
      }

      cat('\n\n\n')
      cat('---------- Other Diagnostic Features -----------\n')

      for(i in diag_props) {
        cat(i,'\n')
        print(diag[[i]],width=width)
        cat('\n')
      }

      cat('\n\n')
      print(orgmin,width=width)
      cat('\n')
      print(org_thk,width=width)
      cat('\n')
      print(partctrl,width=width)
      cat('\n\n\n')
      cat('-- Assumptions/Estimations Due to Missing Inputs --\n')
      for(m in c(1:length(met))) {
```

```
        cat(names(met)[m],'\n')
        print(met[[m]])
        cat('\n')
      }
      cat('\n\n\n\n\n')


      if(!is.na(fn)) {
        sink()
      }







    }


    closeAllConnections()


  }


  rpt <- lapply(ids,function(x){

    report_l(diags=diags, meta=meta, id=x, fn=fn, extended=extended)

  })




}



################################################################################
filter_diags <- function(diags,id) {
  #' A very simple wrapper to make filtering for id cleaner and easier to read
  #' @param diags the main diagnostics data structure
  #' @param id id or ids to filter by

  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  return(diag)

}
```

Appendix C:

calc_taxonomy.R

```
## This script contains the main logic for running taxonomy functions.
##
## Main functions contain 2 modules:
## 1. Main pipe for running all taxonomy with correct logic
## 2. Module for running individual functions by name.
##
##
##



# TODO:
# - conditions not recording for all pedons
calc_taxonomy <- function(hzs, diags, tax='NA', pedon_ids='NA', keys='all',
                          tax_level=NA, ignore.na=FALSE, run.diags=TRUE) {
  #' @param hzs horizons data structure
  #' @param diags diagnostics data structure
  #' @param tax taxonomy data structure
  #' @param pedon_ids vector of unique ids
  #' @param keys vector of specific keys to check
  #' @param tax_level the highest level to go
  #' @param ignore.na flag for ignoring missing values in inputs
  #' @param run.diags flag to run diagnostics or use provided diagnostics



  ## check inputs ##
  checked_inputs <- check_inputs(hzs=hzs, diags=diags, ignore.na=ignore.na)


  ## check tax structure; init if not given ##
  if(class(tax)=="list") {

    samp_tax <- tax_struct(1)
    samp_names <- unlist(lapply(samp_tax,function(x){colnames(x)}))

    in_names <- unlist(lapply(tax,function(x){colnames(x)}))

    if(!identical(samp_names,in_names)) {

      stop("InputError: Input tax structure is not valid.")

    }


  } else {

    ## lengths of inputs must be the same OR a single value

    if(length(pedon_ids)==1 & all(pedon_ids=='NA')) {

      pedon_ids <- unique(hzs$pedon_id)
      ids <- pedon_ids
```

```
    } else {


      if(any(pedon_ids %in% unique(hzs$pedon_id))) {

        ids <- pedon_ids

      } else {

        warning('None of input IDs match data IDs.')

      }

    }

    if(!any(pedon_ids %in% unique(hzs$pedon_id))) {

      stop("InputError: no matching pedon_ids in hzs and tax.")

    }

    tax <- tax_struct(pedon_ids)

  }


  if(run.diags) {

    print("############ Calculating Diagnostics ############")

    diags_calc <-
calc_diagnostics(hzs=checked_inputs[['hzs']],diags=checked_inputs[['diags']],
                                    ignore.na=ignore.na,
                                    pedon_ids=ids)
    hzs <- diags_calc[['hzs']]
    diags <- diags_calc[['diags']]

  } else {

    if(any(is.na(diags$epipedon$hzs))) {
      stop("InputError: provided diagnostics must have epipedon at minimum")
    }

  }
  meta <- checked_inputs[['meta']]




  ##### internal function definition ######
  tax_sect <- function(hzs, diags, tax, id, keys, ord, level,
keys_subset=FALSE) {
```

```
if(length(ord)==1) {

  ords <- rep(ord,length(keys))

} else if(length(ord)!=length(keys)) {

  stop('InputError: orders vector must be the same length as keys vector')

} else {

  ords <- ord
}



if(length(level)==1) {

  lvls <- rep(level,length(keys))

} else if(length(level)!=length(keys)) {

  stop('InputError: level vector must be the same length as keys vector')

} else {

  lvls <- level
}



for(i in c(1:length(keys))) {


  k <- keys[i]
  o <- ords[i]
  l <- lvls[i]

  l <- switch(l,
         order={'orders'},
         suborder={'suborders'},
         greatgroup={'greatgroups'},
         subgroup={'subgroups'},
         l)


  if(l != 'orders') {
    kfun <- paste(c(o,l,k),collapse='.') # convert keys to function names
  } else {
    kfun <- k
  }
```

```
      tryCatch(
        expr={
          f <- match.fun(kfun)
          res <- f(hzs,diags,tax,id)
      },
        error=function(e) {
          msg <- paste(e,kfun,sep=' ')
          stop(msg)
        }
      )

      if(res[[k]]['tax'][res[[k]]['pedon']==id,]) {

        print(k)
        res[['class_tax']][l][res[['class_tax']]['pedon']==id,] <- k
        tax <- res


        if(!keys_subset) {

          break

        }


      } else {

        tax <- res

      }

    }

    return(res)

  }




  # Load ST hierarchy
  # data('ST_logic', package='SoilTaxonomy')
  load(file="C:/Users/Marc/Desktop/NRCS Project/soil_taxonomy/R/soiltaxonomy/
R/data/ST_logic.rda")



  print("############ Calculating Taxonomy ############")



  ############ keys = 'all' ############
```

```
  if('all' %in% keys) {

    t <- lapply(ids,function(id){

      print(paste("Calcualting taxonomy for: ",id,sep=''))

      if(!is.na(tax_level) & (any(tax_level %in% c('orders','suborders','great
groups','subgroups')) |
                              any(tax_level %in% c('order','suborder','great
group','subgroup')))) {
        ## orders ##
        tax <- tax_sect(hzs=hzs, diags=diags, tax=tax, id=id,
keys=unique(ST_logic$order), ord='NA', level='orders')
        ord <- tax[['class_tax']]['orders'][tax[['class_tax']]['pedon']==id,]

        ## run partctrl ##
        if(ord=="histosols") {
          diags <- org_thk(hzs,diags,id)
        } else {
          diags <- partctrl(hzs,diags,tax,id)
        }
        assign(x='diags',value=diags,envir=parent.frame(2))
      }


      if(!is.na(tax_level) & (any(tax_level %in% c('suborders','great
groups','subgroups')) |
                              any(tax_level %in% c('suborder','great
group','subgroup')))) {
        ## suborders ##
        subord_k <- unique(ST_logic$suborder[ST_logic$order==ord]) # keys
        tax <- tax_sect(hzs=hzs, diags=diags, tax=tax, id=id, keys=subord_k,
ord=ord, level='suborders')
        subord <- tax[['class_tax']]['suborders'][tax[['class_tax']]
['pedon']==id,]
      }


      if(!is.na(tax_level) & (any(tax_level %in% c('great
groups','subgroups')) |
                              any(tax_level %in% c('great
group','subgroup')))) {
        ## great groups ##
        gg_k <- unique(ST_logic$greatgroup[ST_logic$suborder==subord]) # keys
        tax <- tax_sect(hzs=hzs, diags=diags, tax=tax, id=id, keys=gg_k,
ord=ord, level='greatgroups')
        gg <- tax[['class_tax']]['greatgroups'][tax[['class_tax']]
['pedon']==id,]
      }

      if(!is.na(tax_level) & (any(tax_level %in% c('subgroups')) |
                              any(tax_level %in% c('subgroup')))) {
        ## subgroups ##
        subg_k <- unique(ST_logic$subgroup[ST_logic$greatgroup==gg]) # keys
```

```
        tax <- tax_sect(hzs=hzs, diags=diags, tax=tax, id=id, keys=subg_k,
ord=ord, level='subgroups')
        subg <- tax[['class_tax']]['subgroups'][tax[['class_tax']]
['pedon']==id,]
      }

      assign(x='tax',value=tax,envir=parent.frame(2))

    })


    objlist <- list(diags=diags,tax=tax,hzs=hzs,meta=meta)
    return(objlist)

  } else {

    ############ keys = subset ############


    # check for valid key inputs
    v <- unlist(lapply(keys,function(x){any(x==ST_logic)}))

    if(!all(v)) {

      i <- keys[!v]
      stop(paste("InvalidKeys: ",paste(i,collapse=', '),sep=""))

    }


    warning('lower taxonomy is built on higher levels; may encounter errors')
    warning('only most recent TRUE for each level will be added to class_tax
when subset')

    t <- lapply(ids,function(id){


      ord <- tolower(unlist(lapply(keys,function(x){

                    l <- SoilTaxonomy::getParentTaxa(x)
                    o <- l[[1]][1]

                    return(o)

                  })))


      level <- unlist(lapply(keys,function(x){

                    n <- which(ST_logic==x,arr.ind=TRUE)[,2][1]
                    lv <- gsub(1,'order',
                               gsub(2,'suborder',
                                    gsub(3,'greatgroup',
                                         gsub(4,'subgroup',n))))
```

```
                  }))


      tax_sect(hzs=hzs, diags=diags, tax=tax, id=id, keys=keys, ord=ord,
level=level, keys_subset=TRUE)



    })

    objlist <- list(diags=diags,tax=tax,hzs=hzs,meta=meta)
    return(objlist)


  }



}




################################################################################
## Function to create report of taxonomy ##
# TODO:
# - consider merging dfs for extended report
# - add definitions to end of report
# - modeled after diagnostics, EDIT
report_taxonomy <- function(tax, ids, fn=NA, extended=FALSE, append=FALSE) {
  #' @param diags diagnostics data structure
  #' @param meta data structure with input assumptions from check_inputs
  #' @param id list of ids
  #' @param fn file name or path to text file
  #' @param extended flag for writing summary report or full report
  #' @param append flag for appending report to existing file


  ##### Taxonomy Names #####
  # ST_logic <- data('ST_logic', package='SoilTaxonomy')
  load(file="C:/Users/Marc/Desktop/NRCS Project/soil_taxonomy/R/soiltaxonomy/
R/data/ST_logic.rda")


  ## Delete old file ##
  if(!append & file.exists(fn)) {
    file.remove(fn)
  }



  ##### Internal Report Function for ID list #####
```

```
  report_t <- function(tax, id, fn, extended) {

    tx <- lapply(tax,function(x){dplyr::filter(x,pedon==id)})



    ########## Basic Summary ##########
    if(!extended) {

      ord <- tx$class_tax$orders
      subord <- tx$class_tax$suborders
      gg <- tx$class_tax$greatgroups
      subg <- tx$class_tax$subgroups



      ##### Write Report ####

      if(!is.na(fn)) {
        sink(file=fn,append=TRUE)
      }

      cat(paste(c('--------------- Taxonomy Report for Pedon
ID:',id,'---------------\n\n'),collapse=' '))
      cat(paste('Order:       ',ord,sep=''),'\n')
      cat(paste('Suborder:    ',subord,sep=''),'\n')
      cat(paste('Great Group: ',gg,sep=''),'\n')
      cat(paste('Subgroup:    ',subg,sep=''),'\n')
      cat('\n\n\n')


      if(!is.na(fn)) {
        sink()
      }

    }




    ########## Extended Summary - Raw output##########
    if(extended) {

      # data('ST_logic', package='SoilTaxonomy')
      load(file="C:/Users/Marc/Desktop/NRCS Project/soil_taxonomy/R/
soiltaxonomy/R/data/ST_logic.rda")

      ord <- tx$class_tax$orders
      subord <- tx$class_tax$suborders
      gg <- tx$class_tax$greatgroups
      subg <- tx$class_tax$subgroups
```

```
      ords <- unique(ST_logic$order)
      subords <- unique(ST_logic$suborder[ST_logic$order==ord])
      ggs <- unique(ST_logic$greatgroup[ST_logic$suborder==subord])
      subgs <- unique(ST_logic$subgroup[ST_logic$greatgroup==gg])



      width <- 300

      if(!is.na(fn)) {
        sink(file=fn,append=TRUE)
      }

      cat(paste(c('----------- Extended Taxonomy Report for Pedon
ID:',id,'-----------\n\n\n'),collapse=' '))
      cat(paste('Order:       ',ord,sep=''),'\n')
      cat(paste('Suborder:    ',subord,sep=''),'\n')
      cat(paste('Great Group: ',gg,sep=''),'\n')
      cat(paste('Subgroup:    ',subg,sep=''),'\n')
      cat('\n\n\n')

      cat('------------------- Orders -------------------\n')
      for(i in ords) {
        cat(i,'\n')
        print(tx[[i]],width=width)
        cat('\n')
      }

      cat('\n\n')

      cat('----------------- Suborders ------------------\n')
      for(i in subords) {
        cat(i,'\n')
        print(tx[[i]],width=width)
        cat('\n')
      }

      cat('\n\n')

      cat('---------------- Great Groups ----------------\n')
      for(i in ggs) {
        cat(i,'\n')
        print(tx[[i]],width=width)
        cat('\n')
      }

      cat('------------------ Subgroups ------------------\n')
      for(i in subgs) {
        cat(i,'\n')
        print(tx[[i]],width=width)
        cat('\n')
      }
```

```
      cat('\n\n\n\n')

      if(!is.na(fn)) {
        sink()
      }



    }


    closeAllConnections()


  }


  rpt <- lapply(ids,function(x){

    report_t(tax=tax, id=x, fn=fn, extended=extended)

  })




}
```

Appendix D:

check_inputs.R

```
## This script contains the main input validation function.
## Input types are validated, conversions are made, and assumptions are made
##
## Checks all inputs in the hzs dataframe.
## Checks inital temperature and moisture regime for the diag struct
##
## Correct units must be used. See documentation.
##
## Returns a list object with hzs and diag



# TODO:
# - check hzs structure
# - combined structure sizes; e.g. 'medium and coarse'
# - reorder to account for new assumptions
# - record any time an assumption is made
# - add check that percentages not over 100
# - spodic materials shortcut for s and h for now
check_inputs <- function(hzs, diags, ignore.na=FALSE) {

  ## Copy original input ##
  h <- hzs

  #### check column names ####

  ## this needs to be updated as needed
  vars <- c("aadipyridyl",
            "acec7",
            "aecec",
            "albic_vol",
            "aloxalate",
            "al_kcl",
            "andic",
            "anhyd",
            "anthraquiccond",
            "aquic",
            "art",
            "bioturb_vol",
            "bs7",
            "bs82",
            "caco3",
            "cec7",
            "cec82",
            "cemagt",
            "cemented",
            "cindfragpum",
            "clay_films",
            "claysizedcarb",
            "claytotal",
            "color_d",
            "color_w",
            "colorchroma_d",
            "colorchroma_w",
```

```
"colorhue_d",
"colorhue_w",
"colorvalue_d",
"colorvalue_w",
"consec_sat",
"cryoturb",
"cumusat",
"dbovendry",
"dbthirdbar",
"densic",
"desgnmaster",
"ec",
"ec_15",
"ecec",
"effclass",
"esp",
"ex_ca",
"ex_mg",
"ex_na",
"extracid",
"failure",
"feoxalate",
"feoxide",
"fragtotvol",
"gelic_mat",
"gypsum",
"humanTpAlt",
"hzdepb",
"hzdept",
"hzname",
"hzthk",
"ksat",
"lamellae_thk",
"lamellae_vol",
"lithdisc",
"lithic",
"melanic_idx",
"midden",
"moss_vol",
"n",
"om",
"oc",
"p_ret",
"paralithic",
"pedon_id",
"permafrost",
"ph1to1h2o",
"ph_cacl2",
"plinthite_vol",
"redoxcolor",
"redoxcolorchroma",
"redoxcolorhue",
"redoxcolorvalue",
"redoxconcen",
```

```
          "redoxconcre",
          "redoxdepl",
          "redoxmass",
          "rll",
          "rrc",
          "sandtotal",
          "sar",
          "sat_mech",
          "seccarb",
          "silt",
          "siltco",
          "slake_h2o",
          "slake_hcl",
          "slake_koh",
          "slake_naoh",
          "sphgm_vol",
          "spodic_mat_pct",
          "structgrade",
          "structsize",
          "structtype",
          "sulfidic_mat",
          "texture",
          "texturemod",
          "vglass",
          "whc",
          "wormhole_vol")


incols <- colnames(hzs)

if(!all(vars %in% incols)) {

  missing_idx <- !(vars %in% incols)
  missing_vars <- vars[missing_idx]

  stop(paste("InputError: hzs columns missing: ",
             paste(missing_vars,collapse=', '),sep=""))

}




## set blanks to NA
hzs[hzs == "" | hzs == " "] <- NA



## order by pedon_id

#### pedon_id ####
```

```
hzs <- hzs[order(hzs$pedon_id), ]



#### hzname ####
tryCatch(
  expr = {

    hzs$hzname <- as.character(hzs$hzname)

    if(any(is.na(hzs$hzname))) {

      stop("InputError: hzname values must be provided.")

    }

  },
  warning = function(e) {

    if(e$message == "NAs introduced by coercion") {

      stop("TypeError: hzname values must be string.")

    }

  }
)


#### desgnmaster ####
tryCatch(
  expr = {

    hzs$desgnmaster <- as.character(hzs$desgnmaster)

    idx <- is.na(hzs$desgnmaster)
    hzs$desgnmaster[idx] <- gsub("[^A-Z]","",hzs$hzname[idx])

  },
  warning = function(e) {

    if(e$message == "NAs introduced by coercion") {

      stop("TypeError: desgnmaster values must be string.")

    }

  }
)


#### al_kcl ####
tryCatch(
  expr = {
```

```
    hzs$al_kcl <- as.numeric(hzs$al_kcl)

    hzs$al_kcl[is.na(hzs$al_kcl)] <- 0


  },
  warning = function(e) {

    if(e$message == "NAs introduced by coercion") {

      stop("TypeError: al_kcl values must be numeric.")

    }



  }
)


#### bs7 ####
tryCatch(
  expr = {

    hzs$bs7 <- as.numeric(hzs$bs7)

    # hzs$bs7[is.na(hzs$bs7)] <- 0


  },
  warning = function(e) {

    if(e$message == "NAs introduced by coercion") {

      stop("TypeError: bs7 values must be numeric.")

    }

  }
)


#### bs82 ####
tryCatch(
  expr = {

    hzs$bs82 <- as.numeric(hzs$bs82)

    # hzs$bs82[is.na(hzs$bs82)] <- 0


  },
  warning = function(e) {
```

```
      if(e$message == "NAs introduced by coercion") {

        stop("TypeError: bs82 values must be numeric.")

      }

    }
  )


  #### claytotal ####
  tryCatch(
    expr = {

      hzs$claytotal <- as.numeric(hzs$claytotal)

      idx1 <- !is.na(hzs$sandtotal) & !is.na(hzs$silt) & is.na(hzs$claytotal)
      hzs$claytotal[idx1] <- 100-hzs$sandtotal[idx1]-hzs$silt[idx1]

      idx2 <- is.na(hzs$claytotal) & !grepl(hzs$desgnmaster,pattern="[ABCE]")
      hzs$claytotal[idx2] <- 0

      if(!ignore.na) {
        if(any(is.na(hzs$claytotal))) {

          stop("InputError: claytotal values must be provided.")

        }
      }

      if(any(hzs$claytotal > 100,na.rm=TRUE)) {

        i <- as.character(hzs$pedon_id[which(hzs$claytotal > 100)])

        stop(paste("InputError: claytotal more than 100% in pedon_id:
",paste(i,collapse=', '),sep=""))

      }


    },
    warning = function(e) {

      if(e$message == "NAs introduced by coercion") {

        stop("TypeError: claytotal values must be numeric.")

      }

    }
  )
```

```
#### cec7 ####
tryCatch(
  expr = {

    hzs$cec7 <- as.numeric(hzs$cec7)

    if(!ignore.na) {
      if(any(is.na(hzs$cec7))) {

        stop("InputError: cec7 values must be provided.")

      }
    }

  },
  warning = function(e) {

    if(e$message == "NAs introduced by coercion") {

      stop("TypeError: cec7 values must be numeric.")

    }

  }
)


#### cec82 ####
tryCatch(
  expr = {

    hzs$cec82 <- as.numeric(hzs$cec82)

    hzs$cec82[is.na(hzs$cec82)] <- 0

  },
  warning = function(e) {

    if(e$message == "NAs introduced by coercion") {

      stop("TypeError: cec82 values must be numeric.")

    }

  }
)

#### ecec ####
tryCatch(
  expr = {

    hzs$ecec <- as.numeric(hzs$ecec)
    idx <- is.na(hzs$ecec) & !is.na(hzs$cec82) & !is.na(hzs$al_kcl)
    hzs$ecec[idx] <- hzs$cec82[idx] + hzs$al_kcl[idx]
```

```
  },
  warning = function(e) {

    if(e$message == "NAs introduced by coercion") {

      stop("TypeError: ecec values must be numeric.")

    }



  }
)



#### aecec ####
tryCatch(
  expr = {

    hzs$aecec <- as.numeric(hzs$aecec)
    idx <- is.na(hzs$aecec) & !is.na(hzs$ecec) & !is.na(hzs$claytotal)
    hzs$aecec[idx] <- hzs$ecec[idx] / (hzs$claytotal[idx]/100)

    if(!ignore.na) {
      if(any(is.na(hzs$aecec))) {

        stop("InputError: aecec values must be provided.")

      }
    }

  },
  warning = function(e) {

    if(e$message == "NAs introduced by coercion") {

      stop("TypeError: aecec values must be numeric.")

    }



  }
)


#### acec7 ####
tryCatch(
  expr = {

    hzs$acec7 <- as.numeric(hzs$acec7)
```

```
    idx <- is.na(hzs$acec7) & !is.na(hzs$cec7)
    hzs$acec7[idx] <- hzs$cec7[idx] / (hzs$claytotal[idx]/100)

    if(!ignore.na) {
      if(any(is.na(hzs$acec7))) {

        stop("InputError: acec7 values must be provided.")

      }
    }

  },
  warning = function(e) {

    if(e$message == "NAs introduced by coercion") {

      stop("TypeError: acec7 values must be numeric.")

    }

  }
)



#### color ####
tryCatch(
  expr = {

    hzs$color_w[!is.na(hzs$color_w)] <- as.character(hzs$color_w)

  },
  warning = function(e) {

    if(e$message == "NAs introduced by coercion") {

      stop("TypeError: color_w values must be string.")

    }

  }
)

tryCatch(
  expr = {

    hzs$color_d[!is.na(hzs$color_d)] <- as.character(hzs$color_d)

  },
  warning = function(e) {

    if(e$message == "NAs introduced by coercion") {

      stop("TypeError: color_d values must be string.")
```

```
      }

    }
  )



  # if wet/dry missing, assume the same as wet/dry
  # idx1 <- is.na(hzs$color_d) & !is.na(hzs$color_w)
  # hzs$color_d[idx1] <- hzs$color_w[idx1]
  #
  # idx2 <- is.na(hzs$color_w) & !is.na(hzs$color_d)
  # hzs$color_w[idx2] <- hzs$color_d[idx2]


  # colors must have a space for this to work
  idx_hue_d <- is.na(hzs$colorhue_d) & !is.na(hzs$color_d)
  idx_hue_w <- is.na(hzs$colorhue_w) & !is.na(hzs$color_w)
  idx_val_d <- is.na(hzs$colorvalue_d) & !is.na(hzs$color_d)
  idx_val_w <- is.na(hzs$colorvalue_w) & !is.na(hzs$color_w)
  idx_chroma_d <- is.na(hzs$colorchroma_d) & !is.na(hzs$color_d)
  idx_chroma_w <- is.na(hzs$colorchroma_w) & !is.na(hzs$color_w)

  hzs$colorhue_d[idx_hue_d] <- unlist(lapply(strsplit(hzs$color_d[idx_hue_d],"
"),function(x){x[1]}))
  hzs$colorhue_w[idx_hue_w] <- unlist(lapply(strsplit(hzs$color_w[idx_hue_w],"
"),function(x){x[1]}))

  hzs$colorvalue_d[idx_val_d] <-
as.numeric(unlist(lapply(strsplit(hzs$color_d[idx_val_d]," "),
                                        function(x)
{lapply(strsplit(x[2],"/"),function(y){y[1]})})))
  hzs$colorvalue_w[idx_val_w] <-
as.numeric(unlist(lapply(strsplit(hzs$color_w[idx_val_w]," "),
                                        function(x)
{lapply(strsplit(x[2],"/"),function(y){y[1]})})))

  hzs$colorchroma_d[idx_chroma_d] <-
as.numeric(unlist(lapply(strsplit(hzs$color_d[idx_chroma_d]," "),
                                        function(x)
{lapply(strsplit(x[2],"/"),function(y){y[2]})})))
  hzs$colorchroma_w[idx_chroma_w] <-
as.numeric(unlist(lapply(strsplit(hzs$color_w[idx_chroma_w]," "),
                                        function(x)
{lapply(strsplit(x[2],"/"),function(y){y[2]})})))



  idx_w_m <- is.na(hzs$colorhue_w) & is.na(hzs$colorvalue_w) &
is.na(hzs$colorchroma_w) &
    !is.na(hzs$colorhue_d) & !is.na(hzs$colorvalue_d) & !
is.na(hzs$colorchroma_d)
```

```
  idx_d_m <- is.na(hzs$colorhue_d) & is.na(hzs$colorvalue_d) &
is.na(hzs$colorchroma_d) &
    !is.na(hzs$colorhue_w) & !is.na(hzs$colorvalue_w) & !
is.na(hzs$colorchroma_w)


  if(any(idx_w_m)) {
    m.moist <- aqp::estimateSoilColor(hzs$colorhue_d[idx_w_m],
                                      hzs$colorvalue_d[idx_w_m],
                                      hzs$colorchroma_d[idx_w_m],
                                      sourceMoistureState = 'dry')

    hzs$colorhue_w[idx_w_m] <- m.moist$hue
    hzs$colorvalue_w[idx_w_m] <- m.moist$value
    hzs$colorchroma_w[idx_w_m] <- m.moist$chroma
    hzs$color_w[idx_w_m] <-
paste(m.moist$hue,paste(m.moist$value,m.moist$chroma,sep='/'),sep=' ')
  }

  if(any(idx_d_m)) {
    m.dry <- aqp::estimateSoilColor(hzs$colorhue_w[idx_d_m],
                                    hzs$colorvalue_w[idx_d_m],
                                    hzs$colorchroma_w[idx_d_m],
                                    sourceMoistureState = 'moist')

    hzs$colorhue_d[idx_d_m] <- m.dry$hue
    hzs$colorvalue_d[idx_d_m] <- m.dry$value
    hzs$colorchroma_d[idx_d_m] <- m.dry$chroma
    hzs$color_d[idx_d_m] <-
paste(m.dry$hue,paste(m.dry$value,m.dry$chroma,sep='/'),sep=' ')

  }



  if(!ignore.na) {
    if(any(is.na(hzs$colorhue_d))) {

      stop("InputError: colorhue_d values must not be NA.")

    }

    if(any(is.na(hzs$colorhue_w))) {

      stop("InputError: colorhue_w values must not be NA.")

    }

    if(any(is.na(hzs$colorvalue_d))) {

      stop("InputError: colorhue_d values must not be NA.")

    }
```

```
  if(any(is.na(hzs$colorvalue_w))) {

    stop("InputError: colorhue_w values must not be NA.")

  }

  if(any(is.na(hzs$colorchroma_d))) {

    stop("InputError: colorchroma_d values must not be NA.")

  }

  if(any(is.na(hzs$colorchroma_w))) {

    stop("InputError: colorchroma_w values must not be NA.")

  }
}



#### dbthirdbar ####
tryCatch(
  expr = {

    hzs$dbthirdbar <- as.numeric(hzs$dbthirdbar)

    if(!ignore.na) {
      if(any(is.na(hzs$dbthirdbar))) {

        stop("InputError: dbthirdbar values must be provided.")

      }
    }

  },
  warning = function(e) {

    if(e$message == "NAs introduced by coercion") {

      stop("TypeError: dbthirdbar values must be numeric.")

    }

  }
)


#### dbovendry ####
tryCatch(
  expr = {

    hzs$dbovendry <- as.numeric(hzs$dbovendry)
```

```
    if(!ignore.na) {
      if(any(is.na(hzs$dbovendry))) {

        stop("InputError: dbovendry values must be provided.")

      }
    }

  },
  warning = function(e) {

    if(e$message == "NAs introduced by coercion") {

      stop("TypeError: dbovendry values must be numeric.")

    }

  }
)


#### ec ####
tryCatch(
  expr = {

    hzs$ec <- as.numeric(hzs$ec)

    hzs$ec[is.na(hzs$ec)] <- 0

  },
  warning = function(e) {

    if(e$message == "NAs introduced by coercion") {

      stop("TypeError: ec values must be numeric.")

    }

  }
)


#### ec_15 ####
tryCatch(
  expr = {

    hzs$ec_15 <- as.numeric(hzs$ec_15)

    hzs$ec_15[is.na(hzs$ec_15)] <- 0


  },
  warning = function(e) {
```

```
    if(e$message == "NAs introduced by coercion") {

      stop("TypeError: ec_15 values must be numeric.")

    }

  }
)


#### ex_ca ####
tryCatch(
  expr = {

    hzs$ex_ca <- as.numeric(hzs$ex_ca)

    hzs$ex_ca[is.na(hzs$ex_ca)] <- 0


  },
  warning = function(e) {

    if(e$message == "NAs introduced by coercion") {

      stop("TypeError: ex_ca values must be numeric.")

    }

  }
)


#### ex_mg ####
tryCatch(
  expr = {

    hzs$ex_mg <- as.numeric(hzs$ex_mg)

    hzs$ex_mg[is.na(hzs$ex_mg)] <- 0

  },
  warning = function(e) {

    if(e$message == "NAs introduced by coercion") {

      stop("TypeError: ex_mg values must be numeric.")

    }

  }
)
```

```
#### ex_na ####
tryCatch(
  expr = {

    hzs$ex_na <- as.numeric(hzs$ex_na)

    hzs$ex_na[is.na(hzs$ex_na)] <- 0


  },
  warning = function(e) {

    if(e$message == "NAs introduced by coercion") {

      stop("TypeError: ex_na values must be numeric.")

    }

  }
)


#### extracid ####
tryCatch(
  expr = {

    hzs$extracid <- as.numeric(hzs$extracid)

    hzs$extracid[is.na(hzs$extracid)] <- 0


  },
  warning = function(e) {

    if(e$message == "NAs introduced by coercion") {

      stop("TypeError: extracid values must be numeric.")

    }

  }
)


#### failure ####
tryCatch(
  expr = {

    hzs$failure <- tolower(as.character(hzs$failure))

    # hzs$failure[hzs$failure == "br"] <- "brittle"
    # hzs$failure[hzs$failure == "sd"] <- "semi-deformable"
    # hzs$failure[hzs$failure == "df"] <- "deformable"
    # hzs$failure[hzs$failure == "nf"] <- "nonfluid"
```

```
    # hzs$failure[hzs$failure == "sf"] <- "slightly fluid"
    # hzs$failure[hzs$failure == "mf"] <- "moderately fluid"
    # hzs$failure[hzs$failure == "vs"] <- "very fluid"
    # hzs$failure[hzs$failure == "ns"] <- "nonsmeary"
    # hzs$failure[hzs$failure == "ws"] <- "weakly smeary"
    # hzs$failure[hzs$failure == "ms"] <- "moderately smeary"
    # hzs$failure[hzs$failure == "sm"] <- "strongly smeary"
    #
    # hzs$failure[is.na(hzs$failure)] <- "brittle"

  },
  warning = function(e) {

    stop("TypeError: failure values must be string.")

  }
)


#### feoxalate ####
tryCatch(
  expr = {

    hzs$feoxalate <- as.numeric(hzs$feoxalate)

    hzs$feoxalate[is.na(hzs$feoxalate)] <- 0


  },
  warning = function(e) {

    if(e$message == "NAs introduced by coercion") {

      stop("TypeError: feoxalate values must be numeric.")

    }

  }
)


#### hzdepb ####
tryCatch(
  expr = {

    hzs$hzdepb <- as.numeric(hzs$hzdepb)

    if(any(is.na(hzs$hzdepb))) {

      stop("InputError: hzdepb values must be provided.")

    }

  },
```

```
  warning = function(e) {

    if(e$message == "NAs introduced by coercion") {

      stop("TypeError: hzdepb values must be numeric.")

    }

  }
)


#### hzdept ####
tryCatch(
  expr = {

    hzs$hzdept <- as.numeric(hzs$hzdept)

    if(any(is.na(hzs$hzdept))) {

      stop("InputError: hzdept values must be provided.")

    }

  },
  warning = function(e) {

    if(e$message == "NAs introduced by coercion") {

      stop("TypeError: hzdept values must be numeric.")

    }

  }
)


#### depth organization and sanity ####
hzs <- hzs[order(ordered(hzs$pedon_id, unique(hzs$pedon_id)), hzs$hzdept), ]

check_overlap <- lapply(unique(hzs$pedon_id),function(x){

  a <- dplyr::filter(hzs,pedon_id==x)
  t <- unlist(mapply(
    FUN = seq,
    from = a[["hzdept"]],
    to = a[["hzdepb"]],
    by = 0.5,
    SIMPLIFY = FALSE
  ))
  b <- t[!t %in% c(a$hzdept,a$hzdepb)]

  if(any(duplicated(b))) {
```

```
      stop(paste("InputError: overlapping horizon bounds in pedon:
",x,sep=""))

    }


  })

  if(!all(hzs$hzdept <= hzs$hzdepb)) {

    idx <- which(hzs$hzdept >= hzs$hzdepb)
    i <- as.character(hzs$pedon_id[idx])
    stop(paste("InputError: hzdept and hzdepb values do not make sense at:
",paste(i,collapse=', '),sep=""))

  }


  #### ksat ####
  tryCatch(
    expr = {

      hzs$ksat <- as.numeric(hzs$ksat)


    },
    warning = function(e) {

      if(e$message == "NAs introduced by coercion") {

        stop("TypeError: ksat values must be numeric.")

      }

    }
  )


  #### whc ####
  tryCatch(
    expr = {

      hzs$whc <- as.numeric(hzs$whc)

    },
    warning = function(e) {

      if(e$message == "NAs introduced by coercion") {

        stop("TypeError: ksat values must be numeric.")

      }

    }
```

```
  )


  #### oc ####
  tryCatch(
    expr = {

      hzs$oc <- as.numeric(hzs$oc)

      hzs$oc[is.na(hzs$oc)] <- 0

      if(any(hzs$oc > 100)) {

        i <- as.character(hzs$pedon_id[which(hzs$oc > 100)])

        stop(paste("InputError: oc more than 100% in pedon_id:
",paste(i,collapse=', '),sep=""))

      }

    },
    warning = function(e) {

      if(e$message == "NAs introduced by coercion") {

        stop("TypeError: oc values must be numeric.")

      }

    }
  )



  #### n ####
  tryCatch(
    expr = {

      hzs$n <- as.numeric(hzs$n)

      idx <- !is.na(hzs$whc) & !is.na(hzs$sandtotal) & !is.na(hzs$claytotal) &
             !is.na(hzs$oc) & is.na(hzs$n)
      hzs$n[idx] <- (as.numeric(hzs$whc[idx]) - 0.2*(100-
as.numeric(hzs$claytotal[idx]))) /
                    (as.numeric(hzs$claytotal[idx]) +
3*(as.numeric(hzs$oc[idx]*1.724)))

      # idx <- !is.na(hzs$whc) & !is.na(hzs$sand) & !is.na(hzs$clay) & !
is.na(hzs$estimated_oc)
      #
      # a <- (as.numeric(hzs$whc[idx])*100 - 0.2*(100-
as.numeric(hzs$clay[idx]))) /
      # (as.numeric(hzs$clay[idx]) +
3*(as.numeric(hzs$estimated_oc[idx])*1.724))
```

```
    if(!ignore.na) {
      if(any(is.na(hzs$n))) {

        stop("InputError: n values must be provided.")

      }
    }

  },
  warning = function(e) {

    if(e$message == "NAs introduced by coercion") {

      stop("TypeError: n values must be numeric.")

    }

  }
)


#### p_ret ####
tryCatch(
  expr = {

    hzs$p_ret <- as.numeric(hzs$p_ret)

    hzs$p_ret[is.na(hzs$p_ret)] <- 0

  },
  warning = function(e) {

    if(e$message == "NAs introduced by coercion") {

      stop("TypeError: p_ret values must be numeric.")

    }

  }
)


#### ph1to1h2o ####
tryCatch(
  expr = {

    hzs$ph1to1h2o <- as.numeric(hzs$ph1to1h2o)


  },
  warning = function(e) {

    if(e$message == "NAs introduced by coercion") {
```

```
      stop("TypeError: ph1to1h2o values must be numeric.")

    }

  }
)


#### ph_cal2 ####
tryCatch(
  expr = {

    hzs$ph_cacl2 <- as.numeric(hzs$ph_cacl2)


  },
  warning = function(e) {

    if(e$message == "NAs introduced by coercion") {

      stop("TypeError: ph_cacl2 values must be numeric.")

    }

  }
)


#### rrc ####
tryCatch(
  expr = {

    hzs$rrc <- tolower(as.character(hzs$rrc))

    hzs$rrc[hzs$rrc == "l"] <- "loose"
    hzs$rrc[hzs$rrc == "s"] <- "soft"
    hzs$rrc[hzs$rrc == "vfr"] <- "very friable"
    hzs$rrc[hzs$rrc == "sh"] <- "slightly hard"
    hzs$rrc[hzs$rrc == "fr"] <- "friable"
    hzs$rrc[hzs$rrc == "mh"] <- "moderately hard"
    hzs$rrc[hzs$rrc == "fi"] <- "firm"
    hzs$rrc[hzs$rrc == "ha"] <- "hard"
    hzs$rrc[hzs$rrc == "vfi"] <- "very firm"
    hzs$rrc[hzs$rrc == "vh"] <- "very hard"
    hzs$rrc[hzs$rrc == "ef"] <- "extremely firm"
    hzs$rrc[hzs$rrc == "eh"] <- "extremely hard"
    hzs$rrc[hzs$rrc == "sr"] <- "slightly rigid"
    hzs$rrc[hzs$rrc == "r"] <- "rigid"
    hzs$rrc[hzs$rrc == "vr"] <- "very rigid"

    if(any(is.na(hzs$rrc))) {

      stop("InputError: rrc values must be provided.")
```

```
      }

    },
    warning = function(e) {

      if(e$message == "NAs introduced by coercion") {

        stop("TypeError: rrc values must be string.")

      }

    }
  )


  #### sandtotal ####
  tryCatch(
    expr = {

      hzs$sandtotal <- as.numeric(hzs$sandtotal)

      idx1 <- is.na(hzs$sandtotal) & !is.na(hzs$silt) & !is.na(hzs$claytotal)
      hzs$sandtotal[idx1] <- 100-hzs$claytotal[idx1]-hzs$silt[idx1]

      idx2 <- is.na(hzs$sandtotal) & !grepl(hzs$desgnmaster,pattern="[ABCE]")
      hzs$sandtotal[idx2] <- 0

      if(!ignore.na) {
        if(any(is.na(hzs$sandtotal))) {

          stop("InputError: sandtotal values must be provided.")

        }
      }

      if(any(hzs$sandtotal > 100,na.rm=TRUE)) {

        i <- as.character(hzs$pedon_id[which(hzs$sandtotal > 100)])

        stop(paste("InputError: sandtotal more than 100% in pedon_id:
",paste(i,collapse=', '),sep=""))

      }

    },
    warning = function(e) {

      if(e$message == "NAs introduced by coercion") {

        stop("TypeError: sandtotal values must be numeric.")

      }
```

```
  }
)


#### structtype ####
tryCatch(
  expr = {

    hzs$structtype <- tolower(as.character(hzs$structtype))

    hzs$structtype[hzs$structtype == "gr"] <- "granular"
    hzs$structtype[hzs$structtype == "abk"] <- "angular blocky"
    hzs$structtype[hzs$structtype == "sbk"] <- "subangular blocky"
    hzs$structtype[hzs$structtype == "lp"] <- "lenticular"
    hzs$structtype[hzs$structtype == "pl"] <- "platy"
    hzs$structtype[hzs$structtype == "weg"] <- "wedge"
    hzs$structtype[hzs$structtype == "pr"] <- "prismatic"
    hzs$structtype[hzs$structtype == "col"] <- "columnar"
    hzs$structtype[hzs$structtype == "sgr"] <- "single grain"
    hzs$structtype[hzs$structtype == "ma"] <- "massive"
    hzs$structtype[hzs$structtype == "cdy"] <- "cloddy"

    hzs$structtype[is.na(hzs$structtype)] <- "massive"

  },
  warning = function(e) {

    if(e$message == "NAs introduced by coercion") {

      stop("TypeError: structtype values must be string.")

    }

  }
)


#### structsize ####
tryCatch(
  expr = {

    hzs$structsize <- tolower(as.character(hzs$structsize))

    hzs$structsize[hzs$structsize == "vf"] <- "very fine"
    hzs$structsize[hzs$structsize == "f"] <- "fine"
    hzs$structsize[hzs$structsize == "m"] <- "medium"
    hzs$structsize[hzs$structsize == "co"] <- "coarse"
    hzs$structsize[hzs$structsize == "vc"] <- "very coarse"
    hzs$structsize[hzs$structsize == "ec"] <- "extremely coarse"

    hzs$structsize[hzs$structsize == "vn"] <- "very thin"
    hzs$structsize[hzs$structsize == "tn"] <- "thin"
    hzs$structsize[hzs$structsize == "tk"] <- "thick"
    hzs$structsize[hzs$structsize == "vk"] <- "very thick"
```

```
      hzs$structsize[hzs$structtype == "massive"] <- NA

    },
    warning = function(e) {

      if(e$message == "NAs introduced by coercion") {

        stop("TypeError: structsize values must be string.")

      }

    }
  )


  #### structgrade ####
  tryCatch(
    expr = {

      hzs$structgrade <- tolower(as.character(hzs$structgrade))

      hzs$structgrade[hzs$structgrade == "0"] <- "structureless"
      hzs$structgrade[hzs$structgrade == "1"] <- "weak"
      hzs$structgrade[hzs$structgrade == "2"] <- "moderate"
      hzs$structgrade[hzs$structgrade == "3"] <- "strong"

      idx <- (hzs$structtype=="massive" | is.na(hzs$structgrade))
      hzs$structgrade[idx] <- "structureless"

    },
    warning = function(e) {

      if(e$message == "NAs introduced by coercion") {

        stop("TypeError: structgrade values must be string.")

      }

    }
  )


  #### sar ####
  tryCatch(
    expr = {

      hzs$sar <- as.numeric(hzs$sar)

      idx1 <- is.na(hzs$sar) & !is.na(hzs$ex_ca) & !is.na(hzs$ex_mg) & !
is.na(hzs$ex_na)
      hzs$sar[idx1] <-  hzs$ex_na[idx1] / (sqrt(0.5*(hzs$ex_ca[idx1] +
hzs$ex_mg[idx1])))
```

```
      idx2 <- is.na(hzs$sar & grepl(hzs$hzname,pattern="[n]"))
      hzs$sar[idx2] <- 13

      hzs$sar[is.na(hzs$sar)] <- 0

    },
    warning = function(e) {

      if(e$message == "NAs introduced by coercion") {

        stop("TypeError: sar values must be numeric.")

      }

    }
  )


  #### esp ####
  tryCatch(
    expr = {

      hzs$esp <- as.numeric(hzs$esp)

      idx <- is.na(hzs$esp) & !is.na(hzs$sar)
      hzs$esp[idx] <- (-0.0126+(0.01475*hzs$sar[idx])) / (1+(-0.0126+
(0.01475*hzs$sar[idx])))

      hzs$esp[idx][hzs$esp[idx]<0] <- 0

      hzs$esp[is.na(hzs$esp)] <- 0

    },
    warning = function(e) {

      if(e$message == "NAs introduced by coercion") {

        stop("TypeError: esp values must be numeric.")

      }

    }
  )


  #### aadipyridyl ####
  tryCatch(
    expr = {

      hzs$aadipyridyl <- as.logical(toupper(hzs$aadipyridyl))

      hzs$aadipyridyl[is.na(hzs$aadipyridyl)] <- FALSE

    },
```

```
  warning = function(e) {

    if(e$message == "NAs introduced by coercion") {

      stop("TypeError: aadipyridyl values must be boolean.")

    }

  }
)


#### albic_vol ####
tryCatch(
  expr = {

    hzs$albic_vol <- as.numeric(hzs$albic_vol)

    hzs$albic_vol[is.na(hzs$albic_vol)] <- 0

  },
  warning = function(e) {

    if(e$message == "NAs introduced by coercion") {

      stop("TypeError: albic_vol values must be numeric.")

    }

  }
)


#### aloxalate ####
tryCatch(
  expr = {

    hzs$aloxalate <- as.numeric(hzs$aloxalate)

    hzs$aloxalate[is.na(hzs$aloxalate)] <- 0

  },
  warning = function(e) {

    if(e$message == "NAs introduced by coercion") {

      stop("TypeError: aloxalate values must be numeric.")

    }

  }
)
```

```
  #### andic ####
  tryCatch(
    expr = {

      hzs$andic <- as.logical(toupper(hzs$andic))

      hzs$andic[is.na(hzs$andic)] <- FALSE

    },
    warning = function(e) {

      if(e$message == "NAs introduced by coercion") {

        stop("TypeError: andic values must be boolean.")

      }

    }
  )


  #### anhyd ####
  tryCatch(
    expr = {

      hzs$anhyd <- as.numeric(hzs$anhyd)

      idx <- is.na(hzs$anhyd) & grepl(hzs$hzname,pattern="aa")

      hzs$anhyd[idx] <- 5

      hzs$anhyd[is.na(hzs$anhyd)] <- 0

      if(any(hzs$anhyd > 100)) {

        i <- as.character(hzs$pedon_id[which(hzs$anhyd > 100)])

        stop(paste("InputError: anhyd more than 100% in pedon_id:
",paste(i,collapse=', '),sep=""))

      }

    },
    warning = function(e) {

      if(e$message == "NAs introduced by coercion") {

        stop("TypeError: anhyd values must be numeric.")

      }

    }
  )
```

```
#### anthraquiccond ####
tryCatch(
  expr = {

    hzs$anthraquiccond <- as.logical(toupper(hzs$anthraquiccond))

    hzs$anthraquiccond[is.na(hzs$anthraquiccond)] <- FALSE

  },
  warning = function(e) {

    if(e$message == "NAs introduced by coercion") {

      stop("TypeError: anthraquiccond values must be boolean.")

    }

  }
)


#### aquic ####
tryCatch(
  expr = {

    hzs$aquic <- as.logical(toupper(hzs$aquic))

    idx1 <- is.na(hzs$aquic) &
      (hzs$redoxconcen==TRUE | hzs$redoxconcre==TRUE | hzs$redoxdepl==TRUE |
         hzs$redoxmass==TRUE | hzs$redoxmass==TRUE)

    hzs$aquic[idx1] <- TRUE

    idx2 <- is.na(hzs$aquic) & grepl(hzs$hzname,pattern="[g]")

    hzs$aquic[idx2] <- TRUE

    hzs$aquic[is.na(hzs$aquic)] <- FALSE

  },
  warning = function(e) {

    if(e$message == "NAs introduced by coercion") {

      stop("TypeError: aquic values must be boolean.")

    }

  }
)


#### art ####
```

```
tryCatch(
  expr = {

    hzs$art <- as.logical(toupper(hzs$art))

    hzs$art[is.na(hzs$art) & grepl(hzs$hzname,pattern="[Uu]")] <- TRUE

    hzs$art[is.na(hzs$art)] <- FALSE

  },
  warning = function(e) {

    if(e$message == "NAs introduced by coercion") {

      stop("TypeError: art values must be boolean.")

    }

  }
)


#### bioturb_vol ####
tryCatch(
  expr = {

    hzs$bioturb_vol <- as.numeric(hzs$bioturb_vol)

    hzs$bioturb_vol[is.na(hzs$bioturb_vol)] <- 0

  },
  warning = function(e) {

    if(e$message == "NAs introduced by coercion") {

      stop("TypeError: bioturb_vol values must be numeric.")

    }

  }
)


#### caco3 ####
tryCatch(
  expr = {

    hzs$caco3 <- as.numeric(hzs$caco3)

    idx_k <- (is.na(hzs$caco3) & grepl(hzs$hzname,pattern="k"))
    hzs$caco3[idx_k] <- 5

    idx_kk <- (is.na(hzs$caco3) & grepl(hzs$hzname,pattern="kk"))
    hzs$caco3[idx_kk] <- 50
```

```
      hzs$caco3[is.na(hzs$caco3)] <- 0

      if(any(hzs$caco3 > 100)) {

        i <- as.character(hzs$pedon_id[which(hzs$caco3 > 100)])

        stop(paste("InputError: caco3 more than 100% in pedon_id:
",paste(i,collapse=', '),sep=""))

      }

    },
    warning = function(e) {

      if(e$message == "NAs introduced by coercion") {

        stop("TypeError: caco3 values must be numeric.")

      }

    }
  )


  #### cemagt ####
  tryCatch(
    expr = {

      hzs$cemagt <- as.character(hzs$cemagt)

      idx_carb <- (is.na(hzs$cemagt) &
                     grepl(hzs$hzname,pattern="[m]") &
                     grepl(hzs$hzname,pattern="[kk]"))
      idx_sil <- (is.na(hzs$cemagt) &
                     grepl(hzs$hzname,pattern="[m]") &
                     grepl(hzs$hzname,pattern="[q]"))
      idx_gyp <- (is.na(hzs$cemagt) &
                    grepl(hzs$hzname,pattern="[m]") &
                    grepl(hzs$hzname,pattern="yy"))
      idx_fe <- (is.na(hzs$cemagt) &
                    grepl(hzs$hzname,pattern="[m]") &
                    grepl(hzs$hzname,pattern="[s]"))
      idx_carb_sil <- (is.na(hzs$cemagt) &
                    grepl(hzs$hzname,pattern="[m]") &
                    grepl(hzs$hzname,pattern="[k]") &
                    grepl(hzs$hzname,pattern="[q]"))
      idx_salt <- (is.na(hzs$cemagt) &
                    grepl(hzs$hzname,pattern="[m]") &
                    grepl(hzs$hzname,pattern="[z]"))
      idx_org <- (is.na(hzs$cemagt) &
                    grepl(hzs$hzname,pattern="[m]") &
                    grepl(hzs$hzname,pattern="[h]"))
```

```
      hzs$cemagt[idx_carb] <- "carbonates"
      hzs$cemagt[idx_sil] <- "silica"
      hzs$cemagt[idx_gyp] <- "gypsum"
      hzs$cemagt[idx_fe] <- "iron"
      hzs$cemagt[idx_carb_sil] <- "carbonates/silca"
      hzs$cemagt[idx_salt] <- "salts"
      hzs$cemagt[idx_org] <- "organic matter"


      idx_unk <- (is.na(hzs$cemagt) &
                    grepl(hzs$hzname,pattern="[m]"))

      hzs$cemagt[idx_unk] <- "unknown"

      hzs$cemagt[is.na(hzs$cemagt)] <- NA

      if(any(is.na(hzs$cemagt) & grepl(hzs$hzname,pattern="[m]"))) {

        stop("InputError: hzname cemented without cementing agent(s) given.")

      }

    },
    warning = function(e) {

      if(e$message == "NAs introduced by coercion") {

        stop("TypeError: cemagt values must be numeric.")

      }

    }
  )


  #### cemented ####
  tryCatch(
    expr = {

      hzs$cemented <- as.logical(toupper(hzs$cemented))

      hzs$cemented[is.na(hzs$cemented) & grepl(hzs$hzname,pattern="[m]")] <-
TRUE

      hzs$cemented[is.na(hzs$cemented)] <- FALSE

    },
    warning = function(e) {

      if(e$message == "NAs introduced by coercion") {

        stop("TypeError: cemented values must be boolean.")

      }
```

```
  }
)


#### cindfragpum ####
tryCatch(
  expr = {

    hzs$cindfragpum <- as.logical(toupper(hzs$cindfragpum))

    hzs$cindfragpum[is.na(hzs$cindfragpum)] <- FALSE

  },
  warning = function(e) {

    if(e$message == "NAs introduced by coercion") {

      stop("TypeError: cindfragpum values must be boolean.")

    }

  }
)


#### clay_films ####
tryCatch(
  expr = {

    hzs$clay_films <- as.logical(toupper(hzs$clay_films))

    # assume Bt horizons have clay films
    idx <- is.na(hzs$clay_films) & grepl(hzs$hzname,pattern="[Tt]")
    hzs$clay_films[idx] <- TRUE
    hzs$clay_films[!idx] <- FALSE

  },
  warning = function(e) {

    if(e$message == "NAs introduced by coercion") {

      stop("TypeError: clay_films values must be boolean.")

    }

  }
)


#### claysizedcarb ####
tryCatch(
  expr = {
```

```
    hzs$claysizedcarb <- as.numeric(hzs$claysizedcarb)

    hzs$claysizedcarb[is.na(hzs$claysizedcarb)] <- 0

  },
  warning = function(e) {

    if(e$message == "NAs introduced by coercion") {

      stop("TypeError: claysiedcarb values must be numeric.")

    }

  }
)


#### consec_sat ####
tryCatch(
  expr = {

    hzs$consec_sat <- as.numeric(hzs$consec_sat)

    hzs$consec_sat[is.na(hzs$consec_sat)] <- 0

  },
  warning = function(e) {

    if(e$message == "NAs introduced by coercion") {

      stop("TypeError: consec_sat values must be numeric.")

    }

  }
)


#### cumusat ####
tryCatch(
  expr = {

    hzs$cumusat <- as.numeric(hzs$cumusat)

    hzs$cumusat[is.na(hzs$cumusat)] <- 0

  },
  warning = function(e) {

    if(e$message == "NAs introduced by coercion") {

      stop("TypeError: cumusat values must be numeric.")

    }
```

```
    }
  )


  #### cryoturb ####
  tryCatch(
    expr = {

      hzs$cryoturb <- as.logical(toupper(hzs$cryoturb))

      hzs$cryoturb[is.na(hzs$cryoturb) & grepl(hzs$hzname,pattern="jj")] <-
TRUE

      hzs$cryoturb[is.na(hzs$cryoturb)] <- FALSE

    },
    warning = function(e) {

      if(e$message == "NAs introduced by coercion") {

        stop("TypeError: cryoturb values must be boolean.")

      }

    }
  )


  #### densic ####
  tryCatch(
    expr = {

      hzs$densic <- as.logical(toupper(hzs$densic))

      # assume Cd is densic
      idx <- is.na(hzs$densic) & grepl(hzs$hzname,pattern="C" ) &
grepl(hzs$hzname,pattern="[Dd]" )
      hzs$densic[idx] <- TRUE
      hzs$densic[is.na(hzs$densic)] <- FALSE

    },
    warning = function(e) {

      if(e$message == "NAs introduced by coercion") {

        stop("TypeError: densic values must be boolean.")

      }

    }
  )
```

```
  ### effclass ####
  tryCatch(
    expr = {

      hzs$effclass <- tolower(as.character(hzs$effclass))

      hzs$effclass[hzs$effclass == "ne"] <- "noneffervescent"
      hzs$effclass[hzs$effclass == "vs"] <- "very slightly effervescent"
      hzs$effclass[hzs$effclass == "sl"] <- "slightly effervescent"
      hzs$effclass[hzs$effclass == "st"] <- "strongly effervescent"
      hzs$effclass[hzs$effclass == "ve"] <- "violently effervescent"

      idx_k <- is.na(hzs$effclass) & !is.na(hzs$cemagt) & (hzs$cemagt ==
"carbonates")
      hzs$effclass[idx_k] <- "strongly effervescent"

      idx_kq <- is.na(hzs$effclass) & !is.na(hzs$cemagt) & (hzs$cemagt ==
"carbonates/silica")
      hzs$effclass[idx_kq] <- "slightly effervescent"

      hzs$effclass[is.na(hzs$effclass)] <- "noneffervescent"

    },
    warning = function(e) {

      if(e$message == "NAs introduced by coercion") {

        stop("TypeError: effclass values must be string.")

      }

    }
  )


  #### feoxide ####
  tryCatch(
    expr = {

      hzs$feoxide <- as.numeric(hzs$feoxide)

      hzs$feoxide[is.na(hzs$feoxide)] <- 0

    },
    warning = function(e) {

      if(e$message == "NAs introduced by coercion") {

        stop("TypeError: feoxide values must be numeric.")

      }

    }
  )
```

```
  #### fragtotvol ####
  tryCatch(
    expr = {

      hzs$fragtotvol <- as.numeric(hzs$fragtotvol)

      hzs$fragtotvol[is.na(hzs$fragtotvol)] <- 0

      if(any(hzs$fragtotvol > 100)) {

        i <- as.character(hzs$pedon_id[which(hzs$fragtotvol > 100)])

        stop(paste("InputError: fragtotvol more than 100% in pedon_id:
",paste(i,collapse=', '),sep=""))

      }

    },
    warning = function(e) {

      if(e$message == "NAs introduced by coercion") {

        stop("TypeError: fragtotvol values must be numeric.")

      }

    }
  )


  #### gelic_mat ####
  tryCatch(
    expr = {

      hzs$gelic_mat <- as.logical(toupper(hzs$gelic_mat))

      hzs$gelic_mat[is.na(hzs$gelic_mat)] <- FALSE

    },
    warning = function(e) {

      if(e$message == "NAs introduced by coercion") {

        stop("TypeError: gelic_mat values must be boolean.")

      }

    }
  )


  #### gypsum ####
```

```
  tryCatch(
    expr = {

      hzs$gypsum <- as.numeric(hzs$gypsum)

      idx_y <- (is.na(hzs$gypsum) & grepl(hzs$hzname,pattern="y"))
      hzs$gypsum[idx_y] <- 5

      idx_yy <- (is.na(hzs$gypsum) & grepl(hzs$hzname,pattern="yy"))
      hzs$gypsum[idx_yy] <- 50

      hzs$gypsum[is.na(hzs$gypsum)] <- 0

      if(any(hzs$gypsum > 100)) {

        i <- as.character(hzs$pedon_id[which(hzs$gypsum > 100)])

        stop(paste("InputError: gypsum more than 100% in pedon_id:
",paste(i,collapse=', '),sep=""))

      }

    },
    warning = function(e) {

      if(e$message == "NAs introduced by coercion") {

        stop("TypeError: gypsum values must be numeric.")

      }

    }
  )


  #### humanTpAlt ####
  tryCatch(
    expr = {

      hzs$humanTpAlt <- as.logical(toupper(hzs$humanTpAlt))

      idx <- is.na(hzs$humanTpAlt) & grepl(hzs$hzname,pattern="\\^")
      hzs$humanTpAlt[idx] <- TRUE

      hzs$humanTpAlt[is.na(hzs$humanTpAlt)] <- FALSE

    },
    warning = function(e) {

      if(e$message == "NAs introduced by coercion") {

        stop("TypeError: humanTpAlt values must be boolean.")

      }
```

```
    }
  )


  #### hzthk ####
  tryCatch(
    expr = {

      hzs$hzthk <- as.numeric(hzs$hzthk)
      hzs$hzthk[is.na(hzs$hzthk)] <- hzs$hzdepb[is.na(hzs$hzthk)] -
hzs$hzdept[is.na(hzs$hzthk)]

      if(any((hzs$hzdepb - hzs$hzdept != hzs$hzthk))) {

        stop("InputError: hzthk values not consistent with boundaries.")

      }


    },
    warning = function(e) {

      if(e$message == "NAs introduced by coercion") {

        stop("TypeError: hzthk values must be numeric.")

      }

    }
  )


  #### lamellae_thk ####
  tryCatch(
    expr = {

      hzs$lamellae_thk <- as.numeric(hzs$lamellae_thk)

      hzs$lamellae_thk[is.na(hzs$lamellae_thk)] <- 0

    },
    warning = function(e) {

      if(e$message == "NAs introduced by coercion") {

        stop("TypeError: lamellae_thk values must be numeric.")

      }

    }
  )
```

```
  #### lamellae_vol ####
  tryCatch(
    expr = {

      hzs$lamellae_vol <- as.numeric(hzs$lamellae_vol)

      hzs$lamellae_vol[is.na(hzs$lamellae_vol)] <- 0

    },
    warning = function(e) {

      if(e$message == "NAs introduced by coercion") {

        stop("TypeError: lamellae_vol values must be numeric.")

      }

    }
  )


  #### lithdisc ####
  tryCatch(
    expr = {

      hzs$lithdisc <- as.logical(toupper(hzs$lithdisc))

      hzs$lithdisc <- unlist(sapply(split(hzs,hzs$pedon_id),

                        function(x){

                          a <- substring(x$hzname, 1, 1) # all first char
                          b <- match(unique(a),a) # index of first unique
                          d <- b[b %in% grep("^[[:digit:]]",a)] # first unique
that are numeric

                          e <- rep(FALSE,length(a))
                          e[d] <- TRUE

                          return(e)

                          }))

      hzs$lithdisc[is.na(hzs$lithdisc)] <- FALSE

    },
    warning = function(e) {

      if(e$message == "NAs introduced by coercion") {

        stop("TypeError: lithdisc values must be boolean.")

      }
```

```
  }
)


#### lithic ####
tryCatch(
  expr = {

    hzs$lithic <- as.logical(toupper(hzs$lithic))

    # assume R is lithic
    idx <- is.na(hzs$lithic) & grepl(hzs$hzname,pattern="[R]" )
    hzs$lithic[idx] <- TRUE
    hzs$lithic[is.na(hzs$lithic)] <- FALSE

  },
  warning = function(e) {

    if(e$message == "NAs introduced by coercion") {

      stop("TypeError: lithic values must be boolean.")

    }

  }
)


#### melanic_idx ####
tryCatch(
  expr = {

    hzs$melanic_idx <- as.numeric(hzs$melanic_idx)

    hzs$melanic_idx[is.na(hzs$melanic_idx)] <- 0

  },
  warning = function(e) {

    if(e$message == "NAs introduced by coercion") {

      stop("TypeError: melanic_idx values must be numeric.")

    }

  }
)


#### midden ####
tryCatch(
  expr = {

    hzs$midden <- as.logical(toupper(hzs$midden))
```

```
      hzs$midden[is.na(hzs$midden)] <- FALSE

    },
    warning = function(e) {

      if(e$message == "NAs introduced by coercion") {

        stop("TypeError: midden values must be boolean.")

      }

    }
  )


  #### moss_vol ####
  tryCatch(
    expr = {

      hzs$moss_vol <- as.numeric(hzs$moss_vol)

      hzs$moss_vol[is.na(hzs$moss_vol)] <- 0

      if(any(hzs$moss_vol > 100)) {

        i <- as.character(hzs$pedon_id[which(hzs$moss_vol > 100)])

        stop(paste("InputError: moss_vol more than 100% in pedon_id:
",paste(i,collapse=', '),sep=""))

      }

    },
    warning = function(e) {

      if(e$message == "NAs introduced by coercion") {

        stop("TypeError: moss_vol values must be numeric.")

      }

    }
  )


  #### paralithic ####
  tryCatch(
    expr = {

      hzs$paralithic <- as.logical(toupper(hzs$paralithic))

      # assume Cr is paralithic
```

```
      idx <- is.na(hzs$paralithic) & grepl(hzs$hzname,pattern="C" ) &
grepl(hzs$hzname,pattern="r" )
      hzs$paralithic[idx] <- TRUE
      hzs$paralithic[is.na(hzs$paralithic)] <- FALSE

    },
    warning = function(e) {

      if(e$message == "NAs introduced by coercion") {

        stop("TypeError: paralithic values must be boolean.")

      }

    }
  )


  #### permafrost ####
  tryCatch(
    expr = {

      hzs$permafrost <- as.logical(toupper(hzs$permafrost))

      hzs$permafrost[is.na(hzs$permafrost) & grepl(hzs$hzname,pattern="[Ff]")]
<- TRUE

      hzs$permafrost[is.na(hzs$permafrost)] <- FALSE

    },
    warning = function(e) {

      if(e$message == "NAs introduced by coercion") {

        stop("TypeError: permafrost values must be boolean.")

      }

    }
  )


  #### plinthite_vol ####
  tryCatch(
    expr = {

      hzs$plinthite_vol <- as.numeric(hzs$plinthite_vol)

      hzs$plinthite_vol[is.na(hzs$plinthite_vol) &
grepl(hzs$hzname,pattern="[v]")] <- 5

      hzs$plinthite_vol[is.na(hzs$plinthite_vol)] <- 0

      if(any(hzs$plinthite_vol > 100)) {
```

```
        i <- as.character(hzs$pedon_id[which(hzs$plinthite_vol > 100)])

        stop(paste("InputError: plinthite_vol more than 100% in pedon_id:
",paste(i,collapse=', '),sep=""))

      }

    },
    warning = function(e) {

      if(e$message == "NAs introduced by coercion") {

        stop("TypeError: plinthite_vol values must be numeric.")

      }

    }
  )


  #### redoxcolor ####
  tryCatch(
    expr = {

      hzs$redoxcolor[!is.na(hzs$redoxcolor)] <- as.character(hzs$redoxcolor)

    },
    warning = function(e) {

      if(e$message == "NAs introduced by coercion") {

        stop("TypeError: redoxcolor values must be string.")

      }

    }
  )


  # colors must have a space for this to work
  idx_hue <- is.na(hzs$redoxcolorhue) & !is.na(hzs$redoxcolor)
  idx_val <- is.na(hzs$redoxcolorvalue) & !is.na(hzs$redoxcolor)
  idx_chroma <- is.na(hzs$redoxcolorchroma) & !is.na(hzs$redoxcolor)


  hzs$redoxcolorhue[idx_hue] <-
unlist(lapply(strsplit(hzs$redoxcolor[idx_hue]," "),function(x){x[1]}))

  hzs$redoxcolorvalue[idx_val] <-
unlist(lapply(strsplit(hzs$redoxcolor[idx_val]," "),
                                                function(x)
{lapply(strsplit(x[2],"/"),function(y){y[1]})}))
```

```
  hzs$redoxcolorchroma[idx_chroma] <-
unlist(lapply(strsplit(hzs$redoxcolor[idx_chroma]," "),
                                                    function(x)
{lapply(strsplit(x[2],"/"),function(y){y[2]})}))



  #### redoxconcen ####
  tryCatch(
    expr = {

      hzs$redoxconcen <- as.logical(toupper(hzs$redoxconcen))

      idx <- !is.na(as.logical(toupper(hzs$redoxconcen))) &
is.na(as.logical(toupper(hzs$redoxconcen)))
      hzs$redoxconcen[idx] <- hzs$redoxconcen[idx]

      hzs$redoxconcen[is.na(hzs$redoxconcen)] <- FALSE

    },
    warning = function(e) {

      if(e$message == "NAs introduced by coercion") {

        stop("TypeError: redoxconcen values must be boolean.")

      }

    }
  )


  #### redoxconcre ####
  tryCatch(
    expr = {

      hzs$redoxconcre <- as.logical(toupper(hzs$redoxconcre))

      hzs$redoxconcre[is.na(hzs$redoxconcre)] <- FALSE

    },
    warning = function(e) {

      if(e$message == "NAs introduced by coercion") {

        stop("TypeError: redoxconcre values must be boolean.")

      }

    }
  )
```

```
  #### redoxdepl ####
  tryCatch(
    expr = {

      hzs$redoxdepl <- as.logical(toupper(hzs$redoxdepl))

      hzs$redoxdepl[is.na(hzs$redoxdepl)] <- FALSE

    },
    warning = function(e) {

      if(e$message == "NAs introduced by coercion") {

        stop("TypeError: redoxdepl values must be boolean.")

      }

    }
  )


  #### redoxmass ####
  tryCatch(
    expr = {

      hzs$redoxmass <- as.logical(toupper(hzs$redoxmass))

      hzs$redoxmass[is.na(hzs$redoxmass)] <- FALSE

    },
    warning = function(e) {

      if(e$message == "NAs introduced by coercion") {

        stop("TypeError: redoxmass values must be boolean.")

      }

    }
  )


  #### rll ####
  ## need to update hzs after calculating diagnostic features
  ## duripan, fragipan, petrocalcic, petrogypsic, placic, densic, lithic,
paralithic, petroferric

  tryCatch(
    expr = {

      hzs$rll <- as.logical(toupper(hzs$rll))

      hzs$rll[is.na(hzs$rll)] <- FALSE
```

```
    },
    warning = function(e) {

      if(e$message == "NAs introduced by coercion") {

        stop("TypeError: rll values must be boolean.")

      }

    }
  )


  #### siltco ####
  tryCatch(
    expr = {

      hzs$siltco <- as.numeric(hzs$siltco)

      hzs$siltco[is.na(hzs$siltco)] <- 0

      if(any(hzs$siltco > 100)) {

        i <- as.character(hzs$pedon_id[which(hzs$siltco > 100)])

        stop(paste("InputError: siltco more than 100% in pedon_id:
",paste(i,collapse=', '),sep=""))

      }

    },
    warning = function(e) {

      if(e$message == "NAs introduced by coercion") {

        stop("TypeError: siltco values must be numeric.")

      }

    }
  )


  #### sat_mech ####
  tryCatch(
    expr = {

      hzs$sat_mech <- toupper(as.character(hzs$sat_mech))

      hzs$sat_mech[is.na(hzs$sat_mech)] <- NA

    },
    warning = function(e) {
```

```
      if(e$message == "NAs introduced by coercion") {

        stop("TypeError: sat_mech values must be character.")

      }

    }
  )


  #### seccarb ####
  tryCatch(
    expr = {

      hzs$seccarb <- as.numeric(hzs$seccarb)

      hzs$seccarb[is.na(hzs$seccarb)] <- 0

      if(any(hzs$seccarb > 100)) {

        i <- as.character(hzs$pedon_id[which(hzs$seccarb > 100)])

        stop(paste("InputError: seccarb more than 100% in pedon_id:
",paste(i,collapse=', '),sep=""))

      }

    },
    warning = function(e) {

      if(e$message == "NAs introduced by coercion") {

        stop("TypeError: seccarb values must be numeric.")

      }

    }
  )


  #### slake_h2o ####
  tryCatch(
    expr = {

      hzs$slake_h2o <- as.numeric(hzs$slake_h2o)

      idx_q <- is.na(hzs$slake_h2o) & !is.na(hzs$cemagt) &
(hzs$cemagt=="silica")
      idx_k <- is.na(hzs$slake_h2o) & !is.na(hzs$cemagt) &
(hzs$cemagt=="carbonates")
      idx_y <- is.na(hzs$slake_h2o) & !is.na(hzs$cemagt) &
(hzs$cemagt=="gypsum")
      idx_s <- is.na(hzs$slake_h2o) & !is.na(hzs$cemagt) &
(hzs$cemagt=="iron")
```

```
      idx_z <- is.na(hzs$slake_h2o) & !is.na(hzs$cemagt) &
(hzs$cemagt=="salts")

      hzs$slake_h2o[idx_q] <- 49
      hzs$slake_h2o[idx_k] <- 51
      hzs$slake_h2o[idx_y] <- 51
      hzs$slake_h2o[idx_s] <- 51
      hzs$slake_h2o[idx_z] <- 51


      hzs$slake_h2o[is.na(hzs$slake_h2o)] <- 100

      if(any(hzs$slake_h2o > 100)) {

        i <- as.character(hzs$pedon_id[which(hzs$slake_h2o > 100)])

        stop(paste("InputError: slake_h2o more than 100% in pedon_id:
",paste(i,collapse=', '),sep=""))

      }

    },
    warning = function(e) {

      if(e$message == "NAs introduced by coercion") {

        stop("TypeError: slake_h2o values must be numeric.")

      }

    }
  )


  #### slake_hcl ####
  tryCatch(
    expr = {

      hzs$slake_hcl <- as.numeric(hzs$slake_hcl)

      idx_q <- is.na(hzs$slake_hcl) & !is.na(hzs$cemagt) &
(hzs$cemagt=="silica")
      idx_k <- is.na(hzs$slake_hcl) & !is.na(hzs$cemagt) &
(hzs$cemagt=="carbonates")
      idx_y <- is.na(hzs$slake_hcl) & !is.na(hzs$cemagt) &
(hzs$cemagt=="gypsum")
      idx_s <- is.na(hzs$slake_hcl) & !is.na(hzs$cemagt) &
(hzs$cemagt=="iron")
      idx_z <- is.na(hzs$slake_hcl) & !is.na(hzs$cemagt) &
(hzs$cemagt=="salts")

      hzs$slake_hcl[idx_q] <- 49
      hzs$slake_hcl[idx_k] <- 51
      hzs$slake_hcl[idx_y] <- 51
```

```
      hzs$slake_hcl[idx_s] <- 51
      hzs$slake_hcl[idx_z] <- 51


      hzs$slake_hcl[is.na(hzs$slake_hcl)] <- 100

      if(any(hzs$slake_hcl > 100)) {

        i <- as.character(hzs$pedon_id[which(hzs$slake_hcl > 100)])

        stop(paste("InputError: slake_hcl more than 100% in pedon_id:
",paste(i,collapse=', '),sep=""))

      }

    },
    warning = function(e) {

      if(e$message == "NAs introduced by coercion") {

        stop("TypeError: slake_hcl values must be numeric.")

      }

    }
  )


  #### slake_koh ####
  tryCatch(
    expr = {

      hzs$slake_koh <- as.numeric(hzs$slake_koh)

      idx_q <- is.na(hzs$slake_hcl) & !is.na(hzs$cemagt) &
(hzs$cemagt=="silica")
      idx_k <- is.na(hzs$slake_hcl) & !is.na(hzs$cemagt) &
(hzs$cemagt=="carbonates")
      idx_y <- is.na(hzs$slake_hcl) & !is.na(hzs$cemagt) &
(hzs$cemagt=="gypsum")
      idx_s <- is.na(hzs$slake_hcl) & !is.na(hzs$cemagt) &
(hzs$cemagt=="iron")
      idx_z <- is.na(hzs$slake_hcl) & !is.na(hzs$cemagt) &
(hzs$cemagt=="salts")

      hzs$slake_hcl[idx_q] <- 51
      hzs$slake_hcl[idx_k] <- 51
      hzs$slake_hcl[idx_y] <- 51
      hzs$slake_hcl[idx_s] <- 51
      hzs$slake_hcl[idx_z] <- 51

      hzs$slake_koh[is.na(hzs$slake_koh)] <- 100

      if(any(hzs$slake_koh > 100)) {
```

```
        i <- as.character(hzs$pedon_id[which(hzs$slake_koh > 100)])

        stop(paste("InputError: slake_koh more than 100% in pedon_id:
",paste(i,collapse=', '),sep=""))

      }

    },
    warning = function(e) {

      if(e$message == "NAs introduced by coercion") {

        stop("TypeError: slake_koh values must be numeric.")

      }

    }
  )


  #### slake_naoh ####
  tryCatch(
    expr = {

      hzs$slake_naoh <- as.numeric(hzs$slake_naoh)

      idx_q <- is.na(hzs$slake_hcl) & !is.na(hzs$cemagt) &
(hzs$cemagt=="silica")
      idx_k <- is.na(hzs$slake_hcl) & !is.na(hzs$cemagt) &
(hzs$cemagt=="carbonates")
      idx_y <- is.na(hzs$slake_hcl) & !is.na(hzs$cemagt) &
(hzs$cemagt=="gypsum")
      idx_s <- is.na(hzs$slake_hcl) & !is.na(hzs$cemagt) &
(hzs$cemagt=="iron")
      idx_z <- is.na(hzs$slake_hcl) & !is.na(hzs$cemagt) &
(hzs$cemagt=="salts")

      hzs$slake_hcl[idx_q] <- 51
      hzs$slake_hcl[idx_k] <- 51
      hzs$slake_hcl[idx_y] <- 51
      hzs$slake_hcl[idx_s] <- 51
      hzs$slake_hcl[idx_z] <- 51

      hzs$slake_naoh[is.na(hzs$slake_naoh)] <- 100

      if(any(hzs$slake_naoh > 100)) {

        i <- as.character(hzs$pedon_id[which(hzs$slake_naoh > 100)])

        stop(paste("InputError: slake_naoh more than 100% in pedon_id:
",paste(i,collapse=', '),sep=""))

      }
```

```
    },
    warning = function(e) {

      if(e$message == "NAs introduced by coercion") {

        stop("TypeError: slake_naoh values must be numeric.")

      }

    }
  )


  #### sphgm_vol ####
  tryCatch(
    expr = {

      hzs$sphgm_vol <- as.numeric(hzs$sphgm_vol)

      hzs$sphgm_vol[is.na(hzs$sphgm_vol)] <- 0

      if(any(hzs$sphgm_vol > 100)) {

        i <- as.character(hzs$pedon_id[which(hzs$sphgm_vol > 100)])

        stop(paste("InputError: sphgm_vol more than 100% in pedon_id:
",paste(i,collapse=', '),sep=""))

      }

    },
    warning = function(e) {

      if(e$message == "NAs introduced by coercion") {

        stop("TypeError: sphgm_vol values must be numeric.")

      }

    }
  )


  #### spodic_mat_pct ####
  tryCatch(
    expr = {

      hzs$spodic_mat_pct <- as.numeric(hzs$spodic_mat_pct)

      ## temporary shortcut
      idx <- is.na(hzs$spodic_mat_pct) & hzs$ph1to1h2o <= 5.9 & hzs$oc >= 0.6
&
             grepl(hzs$hzname,pattern="[sh]")
```

```
      hzs$spodic_mat_pct[idx] <- 100

      hzs$spodic_mat_pct[is.na(hzs$spodic_mat_pct)] <- 0

      if(any(hzs$spodic_mat_pct > 100)) {

        i <- as.character(hzs$pedon_id[which(hzs$spodic_mat_pct > 100)])

        stop(paste("InputError: spodic_mat_pct more than 100% in pedon_id:
",paste(i,collapse=', '),sep=""))

      }

    },
    warning = function(e) {

      if(e$message == "NAs introduced by coercion") {

        stop("TypeError: sphgm_vol values must be numeric.")

      }

    }
  )


  #### sulfidic_mat ####
  tryCatch(
    expr = {

      hzs$sulfidic_mat <- as.logical(toupper(hzs$sulfidic_mat))

      hzs$sulfidic_mat[is.na(hzs$sulfidic_mat)] <- FALSE

    },
    warning = function(e) {

      if(e$message == "NAs introduced by coercion") {

        stop("TypeError: sulfidic_mat values must be boolean.")

      }

    }
  )


  #### texture ####
  tryCatch(
    expr = {

      hzs$texture <- as.character(tolower(hzs$texture))
```

```
      idx <- is.na(hzs$texture) & !is.na(hzs$sandtotal) &
        !is.na(hzs$claytotal) & grepl(hzs$desgnmaster,pattern="[ABCE]")

      if(!all(idx==FALSE)) {

        hzs$texture[idx] <-
as.vector(aqp::ssc_to_texcl(hzs$sandtotal[idx],hzs$claytotal[idx]))

      }


      hzs$texture[hzs$texture == "cos"] <- "coarse sand"
      hzs$texture[hzs$texture == "s"] <- "sand"
      hzs$texture[hzs$texture == "fs"] <- "fine sand"
      hzs$texture[hzs$texture == "vfs"] <- "very sand sand"
      hzs$texture[hzs$texture == "lcos"] <- "loamy coarse sand"
      hzs$texture[hzs$texture == "ls"] <- "loamy sand"
      hzs$texture[hzs$texture == "lfs"] <- "loamy fine sand"
      hzs$texture[hzs$texture == "lvsf"] <- "loamy very fine sand"
      hzs$texture[hzs$texture == "cosl"] <- "coarse sandy loam"
      hzs$texture[hzs$texture == "sl"] <- "sandy loam"
      hzs$texture[hzs$texture == "fsl"] <- "fine sandy loam"
      hzs$texture[hzs$texture == "vfsl"] <- "vfsl"
      hzs$texture[hzs$texture == "l"] <- "loam"
      hzs$texture[hzs$texture == "sil"] <- "silt loam"
      hzs$texture[hzs$texture == "si"] <- "silt"
      hzs$texture[hzs$texture == "scl"] <- "sandy clay loam"
      hzs$texture[hzs$texture == "cl"] <- "clay loam"
      hzs$texture[hzs$texture == "sicl"] <- "silty clay loam"
      hzs$texture[hzs$texture == "sc"] <- "sandy clay"
      hzs$texture[hzs$texture == "sic"] <- "silty clay"
      hzs$texture[hzs$texture == "c"] <- "clay"


    },
    warning = function(e) {

      if(e$message == "NAs introduced by coercion") {

        stop("TypeError: texture values must be string")

      }

    }
  )


  #### texturemod ####
  tryCatch(
    expr = {

      hzs$texturemod <- as.character(tolower(hzs$texturemod))

      idx <- is.na(hzs$texturemod)
```

```
    hzs$texturemod[idx] <- NA

    hzs$texturemod[hzs$texturemod == "gr"] <- "gravelly"
    hzs$texturemod[hzs$texturemod == "grf"] <- "fine gravelly"
    hzs$texturemod[hzs$texturemod == "grm"] <- "medium gravelly"
    hzs$texturemod[hzs$texturemod == "grc"] <- "coarse gravelly"
    hzs$texturemod[hzs$texturemod == "grv"] <- "very gravelly"
    hzs$texturemod[hzs$texturemod == "grx"] <- "extremely gravelly"
    hzs$texturemod[hzs$texturemod == "cb"] <- "cobbly"
    hzs$texturemod[hzs$texturemod == "cbv"] <- "very cobbly"
    hzs$texturemod[hzs$texturemod == "cbx"] <- "extremely cobbly"
    hzs$texturemod[hzs$texturemod == "st"] <- "stony"
    hzs$texturemod[hzs$texturemod == "stv"] <- "very stony"
    hzs$texturemod[hzs$texturemod == "stx"] <- "extremely stony"
    hzs$texturemod[hzs$texturemod == "by"] <- "bouldery"
    hzs$texturemod[hzs$texturemod == "byx"] <- "extremely boulder"
    hzs$texturemod[hzs$texturemod == "cn"] <- "channery"
    hzs$texturemod[hzs$texturemod == "cnv"] <- "very channery"
    hzs$texturemod[hzs$texturemod == "cnx"] <- "extremely channery"
    hzs$texturemod[hzs$texturemod == "fl"] <- "flaggy"
    hzs$texturemod[hzs$texturemod == "flv"] <- "very flaggy"
    hzs$texturemod[hzs$texturemod == "flx"] <- "extremely flaggy"
    hzs$texturemod[hzs$texturemod == "pgr"] <- "paragravelly"
    hzs$texturemod[hzs$texturemod == "pgrv"] <- "very paragravelly"
    hzs$texturemod[hzs$texturemod == "pgrx"] <- "extremely paragrelly"



  },
  warning = function(e) {

    if(e$message == "NAs introduced by coercion") {

      stop("TypeError: texture values must be boolean.")

    }

  }
)


#### vglass ####
tryCatch(
  expr = {

    hzs$vglass <- as.numeric(hzs$vglass)

    hzs$vglass[is.na(hzs$vglass)] <- 0

    if(any(hzs$vglass > 100)) {

      i <- as.character(hzs$pedon_id[which(hzs$vglass > 100)])
```

```
        stop(paste("InputError: vglass more than 100% in pedon_id:
",paste(i,collapse=', '),sep=""))

      }

    },
    warning = function(e) {

      if(e$message == "NAs introduced by coercion") {

        stop("TypeError: vglass values must be numeric.")

      }

    }
  )


  #### wormhole_vol ####
  tryCatch(
    expr = {

      hzs$wormhole_vol <- as.numeric(hzs$wormhole_vol)

      hzs$wormhole_vol[is.na(hzs$wormhole_vol)] <- 0

      if(any(hzs$wormhole_vol > 100)) {

        i <- as.character(hzs$pedon_id[which(hzs$wormhole_vol > 100)])

        stop(paste("InputError: wormhole_vol more than 100% in pedon_id:
",paste(i,collapse=', '),sep=""))

      }

    },
    warning = function(e) {

      if(e$message == "NAs introduced by coercion") {

        stop("TypeError: wormhole_vol values must be numeric.")

      }

    }
  )




  ############# diags ##############
```

```
  ##### SMR #####
  tryCatch(
    expr = {

      diags$smr$smr <- tolower(as.character(diags$smr$smr))

      # SMR not needed for some taxonomy at all

      diags$smr$smr[is.na(diags$smr$smr)] <- 'NA'

      # if(any(is.na(diags$smr$smr))) {
      #
      #   stop("InputError: smr values must be provided.")
      #
      # }

      if(!any(diags$smr$smr %in%
c("aquic","aridic","ustic","udic","xeric","NA"))) {

        stop("InputError: smr must be 'aquic', 'aridic', 'ustic', 'udic', or
'xeric'.")

      }


    },
    warning = function(e) {

      if(e$message == "NAs introduced by coercion") {

        stop("TypeError: smr values must be string.")

      }

    }
  )


  ##### STR #####
  tryCatch(
    expr = {

      diags$str$str <- tolower(as.character(diags$str$str))

      idx <- is.na(diags$str$str) & !is.na(diags$str$mast) & !
is.na(diags$str$msst) & !is.na(diags$str$mwst)
      diags$str$str[idx] <- as.vector(soilDB::estimateSTR(diags$str$mast[idx],
                                                          diags$str$msst[idx],
                                                          diags$str$mwst[idx]))


      if(any(is.na(diags$str$str))) {

        stop("InputError: str values must be provided.")
```

```
      }


      if(!any(diags$str$str %in%
              c("gelic","cryic","frigid","mesic","thermic","hyperthermic",
                "isofrigid","isomesic","isothermic","isohyperthermic"))) {

        stop("InputError: str must be
'gelic','cryic','frigid','mesic','thermic',
             'hyperthermic','isofrigid','isomesic','isothermic','isohyperthermic'.")

      }


    },
    warning = function(e) {

      if(e$message == "NAs introduced by coercion") {

        stop("TypeError: str values must be string.")

      }

    }
  )




  ##### Record Differences #####
  meta <- lapply(unique(hzs$pedon_id),function(x) {

    ho <- dplyr::filter(h, pedon_id==x)
    hz <- dplyr::filter(hzs, pedon_id==x)

    # rnames <- hz$hzname

    dd <- lapply(vars,function(y){

      a <- ho[,y]
      b <- hz[,y]

      idx <- which(is.na(a) & !is.na(b))

      df <- data.frame(ho=a[idx],
                       hz=b[idx])
      colnames(df) <- c('original input','assumed/calc')

      if(nrow(df)!=0) {
        # rownames(df) <- rnames[idx]
        return(df)
      }
```

```
    })

    names(dd) <- vars
    dd <- dd[!unlist(lapply(dd,is.null))]

    return(dd)

  })

  names(meta) <- unique(hzs$pedon_id)



  objlist <- list(hzs=hzs, diags=diags, meta=meta)

  return(objlist)


}
```

Appendix E:

diag_struct.R

```
# outline and framework for diag list of df storage structure
# stores information about all diagnostic features of the pedon
#
# diagnostic horizons, SMR, STR, etc.
# includes depth range where appropriate and/or boolean
#
# passed as argument to diagnostic functions and is updated with results
#


diag_struct <- function(pedon_ids, smr=NA, str=NA, mast=NA, msst=NA, mwst=NA)
{
  #' @param pedon_ids Vector of individual pedon IDs to link to each pedon
being processed
  #' @param smr vector of soil moisture regimes; used to initialize
diagnostics structure
  #' @param str vector of soil temperature regimes; used to initialize
diagnostics structure
  #' @param mast mean annual soil temperature; used to calculate str
  #' @param msst mean summer soil temperature; used to calculate str
  #' @param mwst mean winter soil temperature; used to calculate str
  #' @return Returns diag list of dataframes


  # Epipedons
  anthropic <-
data.frame(pedon=pedon_ids,top_depth=NA,bottom_depth=NA,thick=0,hz=FALSE,
                           cond1=NA,
                           cond2=NA,
                           cond3=NA,
                           cond3a=NA,
                           cond3b1=NA,
                           cond3b2=NA,
                           cond3b3=NA,
                           cond4a=NA,
                           cond4b=NA,
                           cond5=NA)
  folistic <-
data.frame(pedon=pedon_ids,top_depth=NA,bottom_depth=NA,thick=0,hz=FALSE,
                          cond0=NA,
                          cond1=NA,
                          cond1a=NA,
                          cond1b=NA,
                          cond2a=NA,
                          cond2b=NA,
                          cond2c=NA)
  histic <-
data.frame(pedon=pedon_ids,top_depth=NA,bottom_depth=NA,thick=0,hz=FALSE,
                         cond0=NA,
                         cond1=NA,
                         cond1a=NA,
                         cond1b=NA,
                         cond2a=NA,
                         cond2b=NA,
```

```
                        cond2c=NA)
  melanic <-
data.frame(pedon=pedon_ids,top_depth=NA,bottom_depth=NA,thick=0,hz=FALSE,
                         cond1=NA,
                         cond2=NA,
                         cond2a=NA,
                         cond2b=NA,
                         cond2c=NA,
                         cond2d_oc4=NA,
                         cond2d_oc6wa=NA)
  mollic <-
data.frame(pedon=pedon_ids,top_depth=NA,bottom_depth=NA,thick=0,hz=FALSE,
                        cond1a=NA,
                        cond1b=NA,
                        cond2=NA,
                        cond3=NA,
                        cond3a1=NA,
                        cond3a2=NA,
                        cond3b=NA,
                        cond3c=NA,
                        cond4=NA,
                        cond5=NA,
                        cond5a=NA,
                        cond5b=NA,
                        cond5c=NA,
                        cond6a=NA,
                        cond6a1=NA,
                        cond6a2=NA,
                        cond6a3a=NA,
                        cond6a3b=NA,
                        cond6a3=NA,
                        cond6b=NA,
                        cond6c1=NA,
                        cond6c2=NA,
                        cond8=NA)
  umbric <-
data.frame(pedon=pedon_ids,top_depth=NA,bottom_depth=NA,thick=0,hz=FALSE,
                        cond1a=NA,
                        cond1b=NA,
                        cond2=NA,
                        cond3=NA,
                        cond4=NA,
                        cond5a=NA,
                        cond5b=NA,
                        cond6a1=NA,
                        cond6a2=NA,
                        cond6a3=NA,
                        cond6a3a=NA,
                        cond6a3b=NA,
                        cond6b=NA,
                        cond6c1=NA,
                        cond6c2=NA,
                        cond8=NA)
```

```
  ochric <-
data.frame(pedon=pedon_ids,top_depth=NA,bottom_depth=NA,thick=0,hz=FALSE)
  epipedon <-
data.frame(pedon=pedon_ids,top_depth=NA,bottom_depth=NA,thick=0,hz=NA)

  # Diagnostic Subsurface Horizons
  agric <-
data.frame(pedon=pedon_ids,top_depth=NA,bottom_depth=NA,thick=0,hz=FALSE,
                      cond0=NA,
                      cond1=NA,
                      cond2=NA)
  albic <-
data.frame(pedon=pedon_ids,top_depth=NA,bottom_depth=NA,thick=0,hz=FALSE)
  anhydric <-
data.frame(pedon=pedon_ids,top_depth=NA,bottom_depth=NA,thick=0,hz=FALSE,
                         cond1=NA,
                         cond2=NA,
                         cond3=NA)
  argillic <-
data.frame(pedon=pedon_ids,top_depth=NA,bottom_depth=NA,thick=0,hz=FALSE,
                         cond1a=NA,
                         cond1b=NA,
                         cond2=NA)
  calcic <-
data.frame(pedon=pedon_ids,top_depth=NA,bottom_depth=NA,thick=0,hz=FALSE,
                        cond1=NA,
                        cond2=NA,
                        cond2a=NA,
                        cond2b=NA,
                        cond2c=NA,
                        cond2c1=NA,
                        cond2c2=NA,
                        cond2c3=NA,
                        cond3=NA,
                        cond3c=NA)
  cambic <-
data.frame(pedon=pedon_ids,top_depth=NA,bottom_depth=NA,thick=0,hz=FALSE,
                        cond0=NA,
                        cond0b=NA,
                        cond1=NA,
                        cond2a=NA,
                        cond2a1=NA,
                        cond2a2=NA,
                        cond2a3=NA,
                        cond2b=NA,
                        cond2b1=NA,
                        cond3=NA)
  duripan <-
data.frame(pedon=pedon_ids,top_depth=NA,bottom_depth=NA,thick=0,hz=FALSE,
                         cond1=NA,
                         cond3=NA)
  fragipan <-
data.frame(pedon=pedon_ids,top_depth=NA,bottom_depth=NA,thick=0,hz=FALSE,
                         cond0=NA,
```

```
                          cond1=NA,
                          cond3=NA,
                          cond4=NA,
                          cond5=NA,
                          cond6=NA)
  glossic <-
data.frame(pedon=pedon_ids,top_depth=NA,bottom_depth=NA,thick=0,hz=FALSE)
  gypsic <-
data.frame(pedon=pedon_ids,top_depth=NA,bottom_depth=NA,thick=0,hz=FALSE,
                        cond1=NA,
                        cond2=NA,
                        cond3=NA,
                        cond4=NA)
  kandic <-
data.frame(pedon=pedon_ids,top_depth=NA,bottom_depth=NA,thick=0,hz=FALSE,
                        cond1=NA,
                        cond2a=NA,
                        cond2b=NA,
                        cond2b1=NA,
                        cond2b2=NA,
                        cond2b3=NA,
                        cond3a=NA,
                        cond3b=NA,
                        cond4=NA,
                        cond5=NA)
  natric <-
data.frame(pedon=pedon_ids,top_depth=NA,bottom_depth=NA,thick=0,hz=FALSE,
                        cond1a=NA,
                        cond1b=NA,
                        cond2=NA,
                        cond4a=NA,
                        cond5a=NA,
                        cond5b=NA)
  ortstein <-
data.frame(pedon=pedon_ids,top_depth=NA,bottom_depth=NA,thick=0,hz=FALSE,
                         cond1=NA,
                         cond2=NA,
                         cond3=NA)
  oxic <-
data.frame(pedon=pedon_ids,top_depth=NA,bottom_depth=NA,thick=0,hz=FALSE,
                      cond0=NA,
                      cond1=NA,
                      cond2=NA,
                      cond5=NA,
                      cond6=NA)
  petrocalcic <-
data.frame(pedon=pedon_ids,top_depth=NA,bottom_depth=NA,thick=0,hz=FALSE,
                           cond1=NA,
                           cond3a=NA)
  petrogypsic <-
data.frame(pedon=pedon_ids,top_depth=NA,bottom_depth=NA,thick=0,hz=FALSE,
                           cond1=NA,
                           cond3=NA,
                           cond4=NA)
```

```
  placic <-
data.frame(pedon=pedon_ids,top_depth=NA,bottom_depth=NA,thick=0,hz=FALSE,
                       cond1=NA,
                       cond3=NA)
  salic <-
data.frame(pedon=pedon_ids,top_depth=NA,bottom_depth=NA,thick=0,hz=FALSE,
                      cond0=NA,
                      cond1=NA,
                      cond2=NA)
  sombric <-
data.frame(pedon=pedon_ids,top_depth=NA,bottom_depth=NA,thick=0,hz=FALSE)
  spodic <-
data.frame(pedon=pedon_ids,top_depth=NA,bottom_depth=NA,thick=0,hz=FALSE,
                       cond0=NA,
                       cond1=NA)
  sulfuric <-
data.frame(pedon=pedon_ids,top_depth=NA,bottom_depth=NA,thick=0,hz=FALSE)
  glacic <-
data.frame(pedon=pedon_ids,top_depth=NA,bottom_depth=NA,thick=0,hz=FALSE)

  # Other Root Limiting Layers and Contacts
  densic <-
data.frame(pedon=pedon_ids,top_depth=NA,bottom_depth=NA,thick=0,hz=FALSE)
  lithic <-
data.frame(pedon=pedon_ids,top_depth=NA,bottom_depth=NA,thick=0,hz=FALSE)
  paralithic <-
data.frame(pedon=pedon_ids,top_depth=NA,bottom_depth=NA,thick=0,hz=FALSE)
  petroferric <-
data.frame(pedon=pedon_ids,top_depth=NA,bottom_depth=NA,thick=0,hz=FALSE)

  # Pedon-level
  str <- data.frame(pedon=pedon_ids,str=str,mast=mast,msst=msst,mwst=mwst)
  smr <- data.frame(pedon=pedon_ids,smr=smr)

  # General Properties
  abrupt_text <-
data.frame(pedon=pedon_ids,top_depth=NA,bottom_depth=NA,thick=0,prop=FALSE,
                           cond1=NA,
                           cond2=NA,
                           cond2a=NA,
                           cond2b=NA)
  andic_prop <-
data.frame(pedon=pedon_ids,top_depth=NA,bottom_depth=NA,thick=0,prop=FALSE,
                          cond1=NA,
                          cond2=NA,
                          cond2a=NA,
                          cond2b=NA,
                          cond2c=NA,
                          cond3=NA,
                          cond3a=NA,
                          cond3b=NA,
                          cond3c=NA,
                          cond3d=NA,
                          cond3e=NA)
```

```
  albic_mat <-
data.frame(pedon=pedon_ids,top_depth=NA,bottom_depth=NA,thick=0,prop=FALSE,
                           cond1=NA,
                           cond1a=NA,
                           cond1b=NA,
                           cond2=NA,
                           cond2a=NA,
                           cond2b=NA,
                           cond3=NA,
                           cond3a=NA,
                           cond3b=NA)
  anhydrous_cond <-
data.frame(pedon=pedon_ids,top_depth=NA,bottom_depth=NA,thick=0,prop=FALSE)
  fragic_prop <-
data.frame(pedon=pedon_ids,top_depth=NA,bottom_depth=NA,thick=0,prop=FALSE)
  spodic_mat <-
data.frame(pedon=pedon_ids,top_depth=NA,bottom_depth=NA,thick=0,prop=FALSE,
                            cond1=NA,
                            cond2a=NA,
                            cond2a1=NA,
                            cond2a2=NA,
                            cond2a3=NA,
                            cond2a4=NA,
                            cond2b=NA,
                            cond2b1=NA,
                            cond2b3=NA)


  # General Organic or Mineral classification
  orgmin <- data.frame(pedon=pedon_ids,
                       organic_soil=NA,
                       mineral_soil=NA,
                       org_top=NA,
                       org_bottom=NA,
                       min_top=NA,
                       min_bottom=NA)

  org_thk <- data.frame(pedon=pedon_ids,
                        surface_tier_top=NA,
                        surface_tier_bottom=NA,
                        subsurface_tier_top=NA,
                        subsurface_tier_bottom=NA,
                        bottom_tier_top=NA,
                        bottom_tier_bottom=NA,
                        org_ctrl_top=NA,
                        org_ctrl_bottom=NA)
  partctrl <- data.frame(pedon=pedon_ids,
                         pc_top_depth=NA,
                         pc_bottom_depth=NA)


  # Combine as list of dataframes
  diag <- list(
              ### Epipedons ###
```

```
anthropic=anthropic,
folistic=folistic,
histic=histic,
melanic=melanic,
mollic=mollic,
ochric=ochric,
umbric=umbric,
epipedon=epipedon,

### Subsurface Diagnostic ###
agric=agric,
albic=albic,
anhydric=anhydric,
argillic=argillic,
calcic=calcic,
cambic=cambic,
duripan=duripan,
fragipan=fragipan,
glossic=glossic,
gypsic=gypsic,
kandic=kandic,
natric=natric,
ortstein=ortstein,
oxic=oxic,
petrocalcic=petrocalcic,
petrogypsic=petrogypsic,
placic=placic,
salic=salic,
sombric=sombric,
spodic=spodic,
sulfuric=sulfuric,
glacic=glacic,

densic=densic,
lithic=lithic,
paralithic=paralithic,
petroferric=petroferric,

### Temp. & Moisutre ###
str=str,
smr=smr,

### Other Diagnostic ###
abrupt_text=abrupt_text,
andic_prop=andic_prop,
albic_mat=albic_mat,
anhydrous_cond=anhydrous_cond,
fragic_prop=fragic_prop,
spodic_mat=spodic_mat,

orgmin=orgmin,
org_thk=org_thk,
partctrl=partctrl)
```

```
  return(diag)

}
```

Appendix F:

tax_struct.R

```
# outline and framework for taxonomy list of df storage structure
# stores information about all taxonomy conditions
#
# includes depth range where appropriate and/or boolean
#
# passed as argument to taxonomy functions and is updated with results
#


tax_struct <- function(pedon_ids) {
  #' @param pedon_ids Vector of individual pedon IDs to link to each pedon
being processed
  #' @return Returns tax list of dataframes

  ##### Classification #####
  class_tax <- data.frame(pedon=pedon_ids,
                          orders=NA,
                          suborders=NA,
                          greatgroups=NA,
                          subgroups=NA,
                          family=NA)


  ##### Orders #####
  gelisols <- data.frame(pedon=pedon_ids,level='order',tax=FALSE,
                         cond1=NA,
                         cond2=NA)
  histosols <- data.frame(pedon=pedon_ids,level='order',tax=FALSE,
                          cond1=NA,
                          cond2=NA,
                          cond2a=NA,
                          cond2b=NA,
                          cond2c=NA,
                          cond2d=NA,
                          cond2d1=NA,
                          cond2d2=NA)
  spodosols <- data.frame(pedon=pedon_ids,level='order',tax=FALSE,
                          cond0=NA,
                          cond1=NA,
                          cond2=NA,
                          cond3=NA,
                          cond3a=NA,
                          cond3a1=NA,
                          cond3a2=NA,
                          cond3a3=NA,
                          cond3a4=NA,
                          cond3a5=NA,
                          cond3b=NA,
                          cond3b1=NA,
                          cond3b2=NA,
                          cond3c=NA,
                          cond3c1=NA,
                          cond3c2=NA,
                          cond3c2a=NA,
```

```
                        cond3c2b=NA,
                        cond3d=NA,
                        cond3d1=NA,
                        cond3d2=NA,
                        cond3d2a=NA,
                        cond3d2b=NA)
andisols <- data.frame(pedon=pedon_ids,level='order',tax=FALSE,
                       cond1=NA,
                       cond2=NA)
oxisols <- data.frame(pedon=pedon_ids,level='order',tax=FALSE,
                      cond1=NA,
                      cond2=NA)
vertisols <- data.frame(pedon=pedon_ids,level='order',tax=FALSE,
                        cond1=NA,
                        cond2=NA)
aridisols <- data.frame(pedon=pedon_ids,level='order',tax=FALSE,
                        cond1=NA,
                        cond1a=NA,
                        cond1b=NA,
                        cond1c=NA,
                        cond1d=NA,
                        cond2=NA,
                        cond2c=NA)
ultisols <- data.frame(pedon=pedon_ids,level='order',tax=FALSE,
                       cond1=NA,
                       cond1a=NA,
                       cond1a1=NA,
                       cond1a2=NA,
                       cond1b=NA,
                       cond1b1=NA,
                       cond1b2=NA,
                       cond1b3=NA,
                       cond2=NA,
                       cond2a=NA,
                       cond2b=NA,
                       cond2b1=NA,
                       cond2b2=NA,
                       cond2b3=NA)
mollisols <- data.frame(pedon=pedon_ids,level='order',tax=FALSE,
                        cond1=NA,
                        cond1a=NA,
                        cond1b=NA,
                        cond2=NA)
alfisols <- data.frame(pedon=pedon_ids,level='order',tax=FALSE,
                       cond1=NA,
                       cond2=NA)
inceptisols <- data.frame(pedon=pedon_ids,level='order',tax=FALSE,
                          cond1=NA,
                          cond1a=NA,
                          cond1b=NA,
                          cond1c=NA,
                          cond1d=NA,
                          cond1e=NA,
                          cond2=NA,
```

```
                          cond2a=NA,
                          cond2b=NA,
                          cond2b1=NA,
                          cond2b2=NA,
                          cond2b3=NA)
entisols <- data.frame(pedon=pedon_ids,level='order',tax=FALSE)


##### Suborders #####
histels <- data.frame(pedon=pedon_ids,level='suborder',tax=FALSE,
                      cond1=NA,
                      cond2=NA,
                      cond3=NA)
turbels <- data.frame(pedon=pedon_ids,level='suborder',tax=FALSE)
orthels <- data.frame(pedon=pedon_ids,level='suborder',tax=FALSE)

folists <- data.frame(pedon=pedon_ids,level='suborder',tax=FALSE)
wassists <- data.frame(pedon=pedon_ids,level='suborder',tax=FALSE)
fibrists <- data.frame(pedon=pedon_ids,level='suborder',tax=FALSE,
                       cond1=NA,
                       cond1a=NA,
                       cond1b=NA,
                       cond2=NA)
saprists <- data.frame(pedon=pedon_ids,level='suborder',tax=FALSE,
                       cond1=NA,
                       cond2=NA)
hemists <- data.frame(pedon=pedon_ids,level='suborder',tax=FALSE)

aquods <- data.frame(pedon=pedon_ids,level='suborder',tax=FALSE,
                     cond1=NA,
                     cond2=NA)
gelods <- data.frame(pedon=pedon_ids,level='suborder',tax=FALSE)
cryods <- data.frame(pedon=pedon_ids,level='suborder',tax=FALSE)
humods <- data.frame(pedon=pedon_ids,level='suborder',tax=FALSE)
orthods <- data.frame(pedon=pedon_ids,level='suborder',tax=FALSE)

aquands <- data.frame(pedon=pedon_ids,level='suborder',tax=FALSE)
gelands <- data.frame(pedon=pedon_ids,level='suborder',tax=FALSE)
cryands <- data.frame(pedon=pedon_ids,level='suborder',tax=FALSE)
torrands <- data.frame(pedon=pedon_ids,level='suborder',tax=FALSE)
xerands <- data.frame(pedon=pedon_ids,level='suborder',tax=FALSE)
vitrands <- data.frame(pedon=pedon_ids,level='suborder',tax=FALSE)
ustands <- data.frame(pedon=pedon_ids,level='suborder',tax=FALSE)
udands <- data.frame(pedon=pedon_ids,level='suborder',tax=FALSE)

aquox <- data.frame(pedon=pedon_ids,level='suborder',tax=FALSE)
torrox <- data.frame(pedon=pedon_ids,level='suborder',tax=FALSE)
ustox <- data.frame(pedon=pedon_ids,level='suborder',tax=FALSE)
perox <- data.frame(pedon=pedon_ids,level='suborder',tax=FALSE)
udox <- data.frame(pedon=pedon_ids,level='suborder',tax=FALSE)

aquerts <- data.frame(pedon=pedon_ids,level='suborder',tax=FALSE,
                      cond0=NA,
                      cond1=NA)
```

```
cryerts <- data.frame(pedon=pedon_ids,level='suborder',tax=FALSE)
xererts <- data.frame(pedon=pedon_ids,level='suborder',tax=FALSE)
torrerts <- data.frame(pedon=pedon_ids,level='suborder',tax=FALSE)
usterts <- data.frame(pedon=pedon_ids,level='suborder',tax=FALSE)
uderts <- data.frame(pedon=pedon_ids,level='suborder',tax=FALSE)

cryids <- data.frame(pedon=pedon_ids,level='suborder',tax=FALSE)
salids <- data.frame(pedon=pedon_ids,level='suborder',tax=FALSE)
durids <- data.frame(pedon=pedon_ids,level='suborder',tax=FALSE)
gypsids <- data.frame(pedon=pedon_ids,level='suborder',tax=FALSE)
argids <- data.frame(pedon=pedon_ids,level='suborder',tax=FALSE)
calcids <- data.frame(pedon=pedon_ids,level='suborder',tax=FALSE)
cambids <- data.frame(pedon=pedon_ids,level='suborder',tax=FALSE)

aquults <- data.frame(pedon=pedon_ids,level='suborder',tax=FALSE)
humults <- data.frame(pedon=pedon_ids,level='suborder',tax=FALSE,
                      cond1=NA,
                      cond2=NA)
udults <- data.frame(pedon=pedon_ids,level='suborder',tax=FALSE)
ustults <- data.frame(pedon=pedon_ids,level='suborder',tax=FALSE)
xerults <- data.frame(pedon=pedon_ids,level='suborder',tax=FALSE)

albolls <- data.frame(pedon=pedon_ids,level='suborder',tax=FALSE,
                      cond1=NA,
                      cond2=NA,
                      cond3=NA,
                      cond4=NA)
aquolls <- data.frame(pedon=pedon_ids,level='suborder',tax=FALSE,
                      cond0=NA,
                      cond1=NA,
                      cond2=NA,
                      cond3=NA,
                      cond4=NA,
                      cond5=NA,
                      cond5a=NA,
                      cond5a1=NA,
                      cond5a2=NA,
                      cond5a2a=NA,
                      cond5a2b=NA,
                      cond5a2c=NA,
                      cond5a2d=NA,
                      cond5a2e=NA,
                      cond5a2f=NA,
                      cond5b=NA,
                      cond5b1=NA,
                      cond5b2=NA,
                      cond5b2a=NA,
                      cond5b2b=NA,
                      cond5b2c=NA,
                      cond6=NA)
rendolls <- data.frame(pedon=pedon_ids,level='suborder',tax=FALSE,
                       cond1=NA,
                       cond2=NA,
                       cond3=NA,
```

```
                        cond4=NA,
                        cond4a=NA,
                        cond4b=NA)
gelolls <- data.frame(pedon=pedon_ids,level='suborder',tax=FALSE)
cryolls <- data.frame(pedon=pedon_ids,level='suborder',tax=FALSE)
xerolls <- data.frame(pedon=pedon_ids,level='suborder',tax=FALSE)
ustolls <- data.frame(pedon=pedon_ids,level='suborder',tax=FALSE)
ustolls <- data.frame(pedon=pedon_ids,level='suborder',tax=FALSE)
udolls <- data.frame(pedon=pedon_ids,level='suborder',tax=FALSE)

aqualfs <- data.frame(pedon=pedon_ids,level='suborder',tax=FALSE,
                      cond0=NA,
                      cond1=NA,
                      cond1a=NA,
                      cond1b=NA,
                      cond1c=NA,
                      cond2=NA)
cryalfs <- data.frame(pedon=pedon_ids,level='suborder',tax=FALSE)
ustalfs <- data.frame(pedon=pedon_ids,level='suborder',tax=FALSE)
xeralfs <- data.frame(pedon=pedon_ids,level='suborder',tax=FALSE)
udalfs <- data.frame(pedon=pedon_ids,level='suborder',tax=FALSE)


aquepts <- data.frame(pedon=pedon_ids,level='suborder',tax=FALSE,
                      cond1=NA,
                      cond1a=NA,
                      cond1b=NA,
                      cond1c=NA,
                      cond1c1=NA,
                      cond1c2=NA,
                      cond1d=NA,
                      cond2=NA)
gelepts <- data.frame(pedon=pedon_ids,level='suborder',tax=FALSE)
cryepts <- data.frame(pedon=pedon_ids,level='suborder',tax=FALSE)
ustepts <- data.frame(pedon=pedon_ids,level='suborder',tax=FALSE)
xerepts <- data.frame(pedon=pedon_ids,level='suborder',tax=FALSE)
udepts <- data.frame(pedon=pedon_ids,level='suborder',tax=FALSE)

wassents <- data.frame(pedon=pedon_ids,level='suborder',tax=FALSE)
aquents <- data.frame(pedon=pedon_ids,level='suborder',tax=FALSE)
psamments <- data.frame(pedon=pedon_ids,level='suborder',tax=FALSE)
fluvents <- data.frame(pedon=pedon_ids,level='suborder',tax=FALSE,
                       cond0=NA,
                       cond3=NA,
                       cond3a=NA,
                       cond3b=NA,
                       cond4=NA,
                       cond4a=NA,
                       cond4b=NA,
                       cond4b1=NA,
                       cond4b2=NA)
orthents <- data.frame(pedon=pedon_ids,level='suborder',tax=FALSE)
```

```
##### Great Groups #####

## Gelisols ##
folistels <- data.frame(pedon=pedon_ids,level='greatgroup',tax=FALSE)
glacistels <- data.frame(pedon=pedon_ids,level='greatgroup',tax=FALSE,
                         cond0=NA,
                         cond1=NA,
                         cond2=NA)
fibristels <- data.frame(pedon=pedon_ids,level='greatgroup',tax=FALSE)
hemistels <- data.frame(pedon=pedon_ids,level='greatgroup',tax=FALSE)
sapristels <- data.frame(pedon=pedon_ids,level='greatgroup',tax=FALSE)

histoturbels <- data.frame(pedon=pedon_ids,level='greatgroup',tax=FALSE)
aquiturbels <- data.frame(pedon=pedon_ids,level='greatgroup',tax=FALSE)
anhyturbels <- data.frame(pedon=pedon_ids,level='greatgroup',tax=FALSE)
molliturbels <- data.frame(pedon=pedon_ids,level='greatgroup',tax=FALSE)
umbriturbels <- data.frame(pedon=pedon_ids,level='greatgroup',tax=FALSE)
psammoturbels <- data.frame(pedon=pedon_ids,level='greatgroup',tax=FALSE)
haploturbels <- data.frame(pedon=pedon_ids,level='greatgroup',tax=FALSE)

historthels <- data.frame(pedon=pedon_ids,level='greatgroup',tax=FALSE)
aquorthels <- data.frame(pedon=pedon_ids,level='greatgroup',tax=FALSE)
anhyorthels <- data.frame(pedon=pedon_ids,level='greatgroup',tax=FALSE)
mollorthels <- data.frame(pedon=pedon_ids,level='greatgroup',tax=FALSE)
umbrorthels <- data.frame(pedon=pedon_ids,level='greatgroup',tax=FALSE)
argiorthels <- data.frame(pedon=pedon_ids,level='greatgroup',tax=FALSE)
psammorthels <- data.frame(pedon=pedon_ids,level='greatgroup',tax=FALSE)
haplorthels <- data.frame(pedon=pedon_ids,level='greatgroup',tax=FALSE)
histoturbels <- data.frame(pedon=pedon_ids,level='greatgroup',tax=FALSE)

## Histosols ##
cryofibrists <- data.frame(pedon=pedon_ids,level='greatgroup',tax=FALSE)
sphagnofibrists <- data.frame(pedon=pedon_ids,level='greatgroup',tax=FALSE)
haplofibrists <- data.frame(pedon=pedon_ids,level='greatgroup',tax=FALSE)

cryofolists <- data.frame(pedon=pedon_ids,level='greatgroup',tax=FALSE)
torrifolists <- data.frame(pedon=pedon_ids,level='greatgroup',tax=FALSE)
ustifolists <- data.frame(pedon=pedon_ids,level='greatgroup',tax=FALSE)
udifolists <- data.frame(pedon=pedon_ids,level='greatgroup',tax=FALSE)

sulfohemists <- data.frame(pedon=pedon_ids,level='greatgroup',tax=FALSE)
sulfihemists <- data.frame(pedon=pedon_ids,level='greatgroup',tax=FALSE)
luvihemists <- data.frame(pedon=pedon_ids,level='greatgroup',tax=FALSE)
cryohemists <- data.frame(pedon=pedon_ids,level='greatgroup',tax=FALSE)
haplohemists <- data.frame(pedon=pedon_ids,level='greatgroup',tax=FALSE)

sulfosaprists <- data.frame(pedon=pedon_ids,level='greatgroup',tax=FALSE)
sulfisaprists <- data.frame(pedon=pedon_ids,level='greatgroup',tax=FALSE)
cryosaprists <- data.frame(pedon=pedon_ids,level='greatgroup',tax=FALSE)
haplosaprists <- data.frame(pedon=pedon_ids,level='greatgroup',tax=FALSE)

fragiwassists <- data.frame(pedon=pedon_ids,level='greatgroup',tax=FALSE)
sulfiwassists <- data.frame(pedon=pedon_ids,level='greatgroup',tax=FALSE)
haplowassists <- data.frame(pedon=pedon_ids,level='greatgroup',tax=FALSE)
```

```
## Alfisols ##
cryaqualfs <- data.frame(pedon=pedon_ids,level='greatgroup',tax=FALSE)
plinthaqualfs <- data.frame(pedon=pedon_ids,level='greatgroup',tax=FALSE)
duraqualfs <- data.frame(pedon=pedon_ids,level='greatgroup',tax=FALSE)
natraqualfs <- data.frame(pedon=pedon_ids,level='greatgroup',tax=FALSE)
fragiaqualfs <- data.frame(pedon=pedon_ids,level='greatgroup',tax=FALSE)
kandiaqualfs <- data.frame(pedon=pedon_ids,level='greatgroup',tax=FALSE)
vermaqualfs <- data.frame(pedon=pedon_ids,level='greatgroup',tax=FALSE)
albaqualfs <- data.frame(pedon=pedon_ids,level='greatgroup',tax=FALSE)
glossaqualfs <- data.frame(pedon=pedon_ids,level='greatgroup',tax=FALSE)
epiaqualfs <- data.frame(pedon=pedon_ids,level='greatgroup',tax=FALSE)
endoaqualfs <- data.frame(pedon=pedon_ids,level='greatgroup',tax=FALSE)

palecryalfs <- data.frame(pedon=pedon_ids,level='greatgroup',tax=FALSE,
                          cond1=NA,
                          cond1a=NA,
                          cond2=NA,
                          cond3=NA)
glossocryalfs <- data.frame(pedon=pedon_ids,level='greatgroup',tax=FALSE)
haplocryalfs <- data.frame(pedon=pedon_ids,level='greatgroup',tax=FALSE)

natrudalfs <- data.frame(pedon=pedon_ids,level='greatgroup',tax=FALSE)
ferrudalfs <- data.frame(pedon=pedon_ids,level='greatgroup',tax=FALSE)
fraglossudalfs <- data.frame(pedon=pedon_ids,level='greatgroup',tax=FALSE,
                             cond1=NA,
                             cond2=NA)
fragiudalfs <- data.frame(pedon=pedon_ids,level='greatgroup',tax=FALSE)
kandiudalfs <- data.frame(pedon=pedon_ids,level='greatgroup',tax=FALSE,
                          cond1=NA,
                          cond2=NA,
                          cond3=NA,
                          cond3a=NA,
                          cond3b=NA)
kanhapludalfs <- data.frame(pedon=pedon_ids,level='greatgroup',tax=FALSE)
paleudalfs <- data.frame(pedon=pedon_ids,level='greatgroup',tax=FALSE,
                         cond1=NA,
                         cond2=NA,
                         cond2a=NA,
                         cond2b=NA,
                         cond3=NA,
                         cond3a=NA,
                         cond3b=NA,
                         cond3c=NA,
                         cond4=NA,
                         cond4a=NA,
                         cond4a1=NA,
                         cond4b=NA,
                         cond4c=NA)
rhodudalfs <- data.frame(pedon=pedon_ids,level='greatgroup',tax=FALSE,
                        cond0=NA,
                        cond1=NA,
                        cond2=NA,
                        cond3=NA)
```

```
glossudalfs <- data.frame(pedon=pedon_ids,level='greatgroup',tax=FALSE)
hapludalfs <- data.frame(pedon=pedon_ids,level='greatgroup',tax=FALSE)

durustalfs <- data.frame(pedon=pedon_ids,level='greatgroup',tax=FALSE)
plinthustalfs <- data.frame(pedon=pedon_ids,level='greatgroup',tax=FALSE)
natrustalfs <- data.frame(pedon=pedon_ids,level='greatgroup',tax=FALSE)
kandiustalfs <- data.frame(pedon=pedon_ids,level='greatgroup',tax=FALSE,
                           cond1=NA,
                           cond2=NA,
                           cond3=NA,
                           cond3a=NA,
                           cond3b=NA)
kanhaplustalfs <- data.frame(pedon=pedon_ids,level='greatgroup',tax=FALSE)
paleustalfs <- data.frame(pedon=pedon_ids,level='greatgroup',tax=FALSE,
                          cond1=NA,
                          cond2=NA,
                          cond2a=NA,
                          cond2a1=NA,
                          cond2a2=NA,
                          cond2b=NA,
                          cond2b1=NA,
                          cond2b2=NA,
                          cond3=NA,
                          cond3a=NA,
                          cond3b=NA)
rhodustalfs <- data.frame(pedon=pedon_ids,level='greatgroup',tax=FALSE,
                          cond0=NA,
                          cond1=NA,
                          cond2=NA,
                          cond3=NA)
haplustalfs <- data.frame(pedon=pedon_ids,level='greatgroup',tax=FALSE)

durixeralfs <- data.frame(pedon=pedon_ids,level='greatgroup',tax=FALSE)
natrixeralfs <- data.frame(pedon=pedon_ids,level='greatgroup',tax=FALSE)
fragixeralfs <- data.frame(pedon=pedon_ids,level='greatgroup',tax=FALSE)
plinthoxeralfs <- data.frame(pedon=pedon_ids,level='greatgroup',tax=FALSE)
rhodoxeralfs <- data.frame(pedon=pedon_ids,level='greatgroup',tax=FALSE,
                           cond0=NA,
                           cond1=NA,
                           cond2=NA,
                           cond3=NA)
palexeralfs <- data.frame(pedon=pedon_ids,level='greatgroup',tax=FALSE,
                          cond1=NA,
                          cond2=NA,
                          cond2a=NA,
                          cond2a1=NA,
                          cond2a2=NA,
                          cond2b=NA,
                          cond3=NA,
                          cond3a=NA,
                          cond3b=NA)
haploxeralfs <- data.frame(pedon=pedon_ids,level='greatgroup',tax=FALSE)
```

```
# Combine as list of dataframes
tax <- list(
            ## Classification ##
            class_tax=class_tax,
            ## orders ##
            gelisols=gelisols,
            histosols=histosols,
            spodosols=spodosols,
            andisols=andisols,
            oxisols=oxisols,
            vertisols=vertisols,
            aridisols=aridisols,
            ultisols=ultisols,
            mollisols=mollisols,
            alfisols=alfisols,
            inceptisols=inceptisols,
            entisols=entisols,

            ## suborders ##
            histels=histels,
            turbels=turbels,
            orthels=orthels,
            folists=folists,
            wassists=wassists,
            fibrists=fibrists,
            saprists=saprists,
            hemists=hemists,
            aquods=aquods,
            gelods=gelods,
            cryods=cryods,
            humods=humods,
            orthods=orthods,
            aquands=aquands,
            gelands=gelands,
            cryands=cryands,
            torrands=torrands,
            xerands=xerands,
            vitrands=vitrands,
            ustands=ustands,
            udands=udands,
            aquox=aquox,
            torrox=torrox,
            ustox=ustox,
            perox=perox,
            udox=udox,
            aquerts=aquerts,
            cryerts=cryerts,
            xererts=xererts,
            torrerts=torrerts,
            usterts=usterts,
            uderts=uderts,
            cryids=cryids,
            salids=salids,
```

```
durids=durids,
gypsids=gypsids,
argids=argids,
calcids=calcids,
cambids=cambids,
aquults=aquults,
humults=humults,
udults=udults,
ustults=ustults,
xerults=xerults,
albolls=albolls,
aquolls=aquolls,
rendolls=rendolls,
gelolls=gelolls,
cryolls=cryolls,
xerolls=xerolls,
ustolls=ustolls,
udolls=udolls,
aqualfs=aqualfs,
cryalfs=cryalfs,
ustalfs=ustalfs,
xeralfs=xeralfs,
udalfs=udalfs,
aquepts=aquepts,
gelepts=gelepts,
cryepts=cryepts,
ustepts=ustepts,
xerepts=xerepts,
udepts=udepts,
wassents=wassents,
aquents=aquents,
psamments=psamments,
fluvents=fluvents,
orthents=orthents,

## Great Groups ##
# Gelisols #
folistels=folistels,
glacistels=glacistels,
fibristels=fibristels,
hemistels=hemistels,
sapristels=sapristels,
histoturbels=histoturbels,
aquiturbels=aquiturbels,
anhyturbels=anhyturbels,
molliturbels=molliturbels,
umbriturbels=umbriturbels,
psammoturbels=psammoturbels,
haploturbels=haploturbels,
historthels=historthels,
aquorthels=aquorthels,
anhyorthels=anhyorthels,
mollorthels=mollorthels,
umbrorthels=umbrorthels,
```

```
argiorthels=argiorthels,
psammorthels=psammorthels,
haplorthels=haplorthels,

# Histosols #
cryofibrists=cryofibrists,
sphagnofibrists=sphagnofibrists,
haplofibrists=haplofibrists,
cryofolists=cryofolists,
torrifolists=torrifolists,
ustifolists=ustifolists,
udifolists=udifolists,
sulfohemists=sulfohemists,
sulfihemists=sulfihemists,
luvihemists=luvihemists,
cryohemists=cryohemists,
haplohemists=haplohemists,
sulfosaprists=sulfosaprists,
sulfisaprists=sulfisaprists,
cryosaprists=cryosaprists,
haplosaprists=haplosaprists,
frasiwassists=fragiwassists,
sulfiwassists=sulfiwassists,
haplowassists=haplowassists,

# Alfisols #
cryaqualfs=cryaqualfs,
plinthaqualfs=plinthaqualfs,
duraqualfs=duraqualfs,
natraqualfs=natraqualfs,
fragiaqualfs=fragiaqualfs,
kandiaqualfs=kandiaqualfs,
vermaqualfs=vermaqualfs,
albaqualfs=albaqualfs,
glossaqualfs=glossaqualfs,
epiaqualfs=epiaqualfs,
endoaqualfs=endoaqualfs,
palecryalfs=palecryalfs,
glossocryalfs=glossocryalfs,
haplocryalfs=haplocryalfs,
natrudalfs=natrudalfs,
ferrudalfs=ferrudalfs,
fraglossudalfs=fraglossudalfs,
fragiudalfs=fragiudalfs,
kandiudalfs=kandiudalfs,
kanhapludalfs=kanhapludalfs,
paleudalfs=paleudalfs,
rhodudalfs=rhodudalfs,
glossudalfs=glossudalfs,
hapludalfs=hapludalfs,
durustalfs=durustalfs,
plinthustalfs=plinthustalfs,
natrustalfs=natrustalfs,
kandiustalfs=kandiustalfs,
```

```
                kanhaplustalfs=kanhaplustalfs,
                paleustalfs=paleustalfs,
                rhodustalfs=rhodustalfs,
                haplustalfs=haplustalfs,
                durixeralfs=durixeralfs,
                natrixeralfs=natrixeralfs,
                fragixeralfs=fragixeralfs,
                plinthoxeralfs=plinthoxeralfs,
                rhodoxeralfs=rhodoxeralfs,
                palexeralfs=palexeralfs,
                haploxeralfs=haploxeralfs



                 )

  return(tax)


}
```

Appendix G:

epipedons.R

```
# This script contains functions for walking through the logic
# for all epipedons from the Keys to Soil Taxonomy
#
# Input data corresponds to actual inputs needed for each epipedon.
# The raw inputs (e.g. diagnostic features) may need to be calculated
# outside of these functions beforehand.
#
#

# TODO:
# - Add skipped conditions; e.g. secondary carbonates
# - Add Plaggen
# - Revise code so all epipedons have same "style"


## Epipedon functions ##

##############################################################################
epi.anthropic <- function(hzs, diags, id) {
  #' Function to walk through logic to determine if epipedon is Anthropic
  #'
  #' Parameters and var names are temporary defendant on input data structure
  #' hz input will likely be df of horizons with horizon-level data which
  #' should include structure, rock fragments, n values, etc.
  #'
  #'
  #' hz df requires values for:
  #' pedon_id
  #' structtype - structure; e.g. Granular, Prismastic etc.
  #' structsize - size of aggregates: e.g. Fine, Medium, Coarse, etc.
  #' fragtotvol - total vol of rock frags as %
  #' humanTpAlt - human altered or transported material; TRUE or FALSE
  #' lithdisc   - lithologic discontinuity; TRUE or FALSE
  #' rll        - root limiting layer; TRUE or FALSE
  #' art  - artifact total volume; as %
  #' midden     - midden; TRUE or FALSE
  #' n_val      - n value
  #' hzdept
  #' hzdepb


  hz <- dplyr::filter(hzs, pedon_id==id)
  #idx <- which(hz$pedon_id==id)

  # Section: 1 - Aggregates must be <= 30cm
  # cond1 <- which(
  #   hz$structtype[idx] %in% c("Granular","Platy") |
  #   (hz$structtype[idx] %in% c("Lenticular","Angular Blocky","Subangular
Blocky") & hz$structsize[idx] != "Very Coarse") |
  #   (hz$structtype[idx] %in% c("Prismatic","Columnar","Wedge") &
hz$structsize[idx] %in% c("Very Fine","Fine","Medium"))
  #   )

  cond1 <- which(
```

```
    tolower(hz$structtype) %in% c("granular","platy") |
      (tolower(hz$structtype) %in% c("lenticular","angular blocky","subangular
blocky") & hz$structsize != "very coarse") |
      (tolower(hz$structtype) %in% c("prismatic","columnar","wedge") &
hz$structsize %in% c("very fine","fine","medium"))
  )

  cond1consec <- rle(diff(cond1))

  if(!any(cond1consec$lengths>=1 & cond1consec$values==1 & 1 %in% cond1)) {
    diags[["anthropic"]]["hz"][diags[["anthropic"]]["pedon"]==id,] <- FALSE
    diags[["anthropic"]]["cond1"][diags[["anthropic"]]["pedon"]==id,] <- FALSE
    return(diags)
  } else {
    diags[["anthropic"]]["cond1"][diags[["anthropic"]]["pedon"]==id,] <- TRUE
  }

  idx <- split(cond1, cumsum(c(1, diff(cond1) != 1)))[[1]]



  # Section: 2 - Rock frags < 0.5 Volume
  # assumed col with total rock frags by volume
  cond2 <- which(hz$fragtotvol[idx] < 50)
  cond2consec <- rle(diff(cond2))

  if(!any(cond2consec$lengths>=1 & cond2consec$values==1 & 1 %in% cond2)) {
    diags[["anthropic"]]["hz"][diags[["anthropic"]]["pedon"]==id,] <- FALSE
    diags[["anthropic"]]["cond2"][diags[["anthropic"]]["pedon"]==id,] <- FALSE
    return(diags)
  } else {
    diags[["anthropic"]]["cond2"][diags[["anthropic"]]["pedon"]==id,] <- TRUE
  }

  idx <- split(cond2, cumsum(c(1, diff(cond2) != 1)))[[1]]



  # Section: 3 - Human-altered or transported material
  # assumed col with boolean humanTpAlt material
  cond3 <- which(hz$humanTpAlt[idx] == TRUE)
  cond3consec <- rle(diff(cond3))

  if(!any(cond3consec$lengths>=1 & cond3consec$values==1 & 1 %in% cond3)) {
    diags[["anthropic"]]["hz"][diags[["anthropic"]]["pedon"]==id,] <- FALSE
    diags[["anthropic"]]["cond3"][diags[["anthropic"]]["pedon"]==id,] <- FALSE
    return(diags)
  } else {
    diags[["anthropic"]]["cond3"][diags[["anthropic"]]["pedon"]==id,] <- TRUE
  }

  idx <- split(cond3, cumsum(c(1, diff(cond3) != 1)))[[1]]
```

```
  # Section: 3a - Anthropic Epipedon overlays spodic material and RLL
  # or lith discont that is NOT human transport or altered
  # assumes column for lithologic discontinuity and RLL
  # assumes column for artifact volume

  underlay <- max(idx)+1

  if(hz$pedon_id[underlay]==id & hz$humanTpAlt[underlay]==FALSE &
     (hz$lithdisc[underlay] | hz$rll[underlay]) & hz$fragtotvol[underlay]>0) {

    cond3a <- TRUE


  } else if(hz$art[min(idx)]) {
    idx <- split(idx, cumsum(c(1, diff(idx) != 1)))[[1]]
    cond3b1 <- TRUE
    diags[["anthropic"]]["cond3b1"][diags[["anthropic"]]["pedon"]==id,] <-
TRUE
  } else if(hz$midden[underlay]==TRUE) {
    cond3b2 <- TRUE
    diags[["anthropic"]]["cond3b2"][diags[["anthropic"]]["pedon"]==id,] <-
TRUE
  } else if(diags$anthraquiccond[which(diags$pedon_id==id)]==TRUE) {
    cond3b3 <- TRUE
    diags[["anthropic"]]["cond3b3"][diags[["anthropic"]]["pedon"]==id,] <-
TRUE
  } else {
    diags[["anthropic"]]["hz"][diags[["anthropic"]]["pedon"]==id,] <- FALSE
    diags[["anthropic"]]["cond3b1"][diags[["anthropic"]]["pedon"]==id,] <-
FALSE
    diags[["anthropic"]]["cond3b2"][diags[["anthropic"]]["pedon"]==id,] <-
FALSE
    diags[["anthropic"]]["cond3b3"][diags[["anthropic"]]["pedon"]==id,] <-
FALSE
  }


  # Anthropic epipedon extends to the top of the first
  # underlaying diagnostic illuvial horizon (argillic, kandic, natric or
spodic)
  # assumes diags df has checked for sanity; e.g.
  # Section: 0

  diag_illuv <- diags[c("argillic","kandic","natric","spodic")]

  if(any(as.logical(lapply(diag_illuv,function(x){x[["hz"]]
[which(x[["pedon"]]==id)]})))) {
    diags[["anthropic"]]["top_depth"][diags[["anthropic"]]["pedon"]==id,] <-
hz$hzdept[1]
    diags[["anthropic"]]["bottom_depth"][diags[["anthropic"]]["pedon"]==id,]
<- diags[[which(lapply(diags,function(x){x[["hz"]]
[which(x[["pedon"]]==id)]})==TRUE)]][[2]]
```

```
    diags[["anthropic"]]["thick"][diags[["anthropic"]]["pedon"]==id,] <-
diags[[which(lapply(diags,function(x){x[["hz"]]
[which(x[["pedon"]]==id)]})==TRUE)]][[2]]

  }

  # Section 4a
  if(any(hz$rll) & min(hz$hzdept[which(hz$rll)],na.rm=TRUE) < 25) {
    diags[["anthropic"]]["top_depth"][diags[["anthropic"]]["pedon"]==id,] <-
hz$hzdept[1]
    diags[["anthropic"]]["bottom_depth"][diags[["anthropic"]]["pedon"]==id,]
<- min(hz$hzdept[which(hz$rll)],na.rm=TRUE)
    diags[["anthropic"]]["thick"][diags[["anthropic"]]["pedon"]==id,] <-
min(hz$hzdept[which(hz$rll)],na.rm=TRUE)
    diags[["anthropic"]]["cond4a"][diags[["anthropic"]]["pedon"]==id,] <- TRUE
  }

  #Section 4b
  else if(hz$hzdepb[max(idx)] >= 25) {
    diags[["anthropic"]]["cond4b"][diags[["anthropic"]]["pedon"]==id,] <- TRUE
    weights_25 <- depthWeights(hz$hzdept[idx],hz$hzdepb[idx],0,25)
    n_val_25 <- sum(hz$n[idx]*weights_25)

    # section 5
    if(n_val_25 < 0.7){
      diags[["anthropic"]]["top_depth"][diags[["anthropic"]]["pedon"]==id,] <-
hz$hzdept[1]
      diags[["anthropic"]]["bottom_depth"][diags[["anthropic"]]["pedon"]==id,]
<- 25
      diags[["anthropic"]]["thick"][diags[["anthropic"]]["pedon"]==id,] <- 25
      diags[["anthropic"]]["cond5"][diags[["anthropic"]]["pedon"]==id,] <-
TRUE
      diags[["anthropic"]]["hz"][diags[["anthropic"]]["pedon"]==id,] <- TRUE
    } else {
      diags[["anthropic"]]["hz"][diags[["anthropic"]]["pedon"]==id,] <- FALSE
      diags[["anthropic"]]["cond5"][diags[["anthropic"]]["pedon"]==id,] <-
FALSE
    }

  }


  if(diags[["anthropic"]]["hz"][diags[["anthropic"]]["pedon"]==id,]) {
    # Anthropic epipedon extends to the top of the first
    # underlaying diagnostic illuvial horizon (argillic, kandic, natric or
spodic)
    # assumes diags df has checked for sanity; e.g.
    # Section: 0

    diag_illuv <- diags[c("argillic","kandic","natric","spodic")]

    if(any(as.logical(lapply(diag_illuv,function(x){x[["hz"]]
[which(x[["pedon"]]==id)]})))) {
```

```
      diags[["anthropic"]]["top_depth"][diags[["anthropic"]]["pedon"]==id,] <-
hz$hzdept[1]
      diags[["anthropic"]]["bottom_depth"][diags[["anthropic"]]["pedon"]==id,]
<- diags[[which(lapply(diags,function(x){x[["hz"]]
[which(x[["pedon"]]==id)]})==TRUE)]][[2]]
      diags[["anthropic"]]["thick"][diags[["anthropic"]]["pedon"]==id,] <-
diags[["anthropic"]]["bottom_depth"][diags[["anthropic"]]["pedon"]==id,]

    }

  }
  return(diags)

}









################################################################################
# TODO:
# - changed bd from 1/3 bar to oven dry
epi.folistic <- function(hzs, diags, id) {


  hz <- dplyr::filter(hzs, pedon_id==id)


  cond0 <- all(hz$cumusat < 30)

  if(!cond0) {

    diags[["folistic"]]["hz"][diags[["folistic"]]["pedon"]==id,] <- FALSE
    diags[["folistic"]]["cond0"][diags[["folistic"]]["pedon"]==id,] <- FALSE
    return(diags)

  }



  # depth of the organic portion
  cond1 <- dplyr::filter(diags[["orgmin"]],pedon==id)[["org_bottom"]]



  if(is.na(cond1)) {

    diags[["folistic"]]["hz"][diags[["folistic"]]["pedon"]==id,] <- FALSE
```

```
    diags[["folistic"]]["cond1"][diags[["folistic"]]["pedon"]==id,] <- FALSE
    return(diags)

  }



  # Cond 1a: sphagnum and Db
  if(hz$sphgm_vol[1] > 0) {
    idx <- which(hz$sphgm_vol > 0)
    sphgm_idx <- split(idx, cumsum(c(1, diff(idx) != 1)))[[1]]
    v <- hz$sphgm_vol[sphgm_idx]
    t <- hz$hzdepb[sphgm_idx] - hz$hzdept[sphgm_idx]
    ft <- hz$hzdepb[max(sphgm_idx)]
    w <- t/ft
    sphgm_vol <- weighted.mean(v,w)

    if(ft >= 20) {
      s <- seq(21,ft,1)
      vols <- roll_wa(hz=hz, s=s, prop="sphgm_vol")
      cond1a_sphgm <- max(which(vols >= 75)) # index
      if(length(cond1a_sphgm)==0) {
        cond1a_sphgm <- FALSE

      } else {

        cond1a <- TRUE
        diags[["folistic"]]["cond1a"][diags[["folistic"]]["pedon"]==id,] <-
TRUE
        diags[["folistic"]]["top_depth"][diags[["folistic"]]["pedon"]==id,] <-
hz$hzdept[1]
        diags[["folistic"]]["bottom_depth"][diags[["folistic"]]["pedon"]==id,]
<- s[cond1a_sphgm]
        diags[["folistic"]]["thick"][diags[["folistic"]]["pedon"]==id,] <-
s[cond1a_sphgm]
        diags[["folistic"]]["hz"][diags[["folistic"]]["pedon"]==id,] <- TRUE

      }

    } else if(ft==20 & sphgm_vol < 75) {

      cond1a_sphgm <- FALSE


    } else {

      cond1a_sphgm <- FALSE

    }


  } else {

    cond1a_sphgm <- FALSE
```

```
  }


  # if sphagnum criteria next met, check bulk density criteria
  if(cond1a_sphgm == FALSE & !is.na(hz$dbovendry[1]) & hz$dbovendry[1] < 0.1 )
{

    idx <- which(hz$dbovendry <= 0.1)
    db_idx <- split(idx, cumsum(c(1, diff(idx) != 1)))[[1]]

    db <- hz$dbovendry[db_idx]
    t <- hz$hzdepb[db_idx] - hz$hzdept[db_idx]
    ft <- hz$hzdepb[max(db_idx)]
    w <- t/ft
    db_wt <- weighted.mean(db,w)

    if(ft>=20) {
      s <- seq(21,ft,1)
      dbs <- roll_wa(hz=hz, s=s, prop="dbovendry")
      cond1a_db <- max(which(dbs >= 75)) # index
      if(length(cond1a_db)==0) {
        cond1a <- FALSE

      } else {

        cond1a <- TRUE
        diags[["folistic"]]["cond1a"][diags[["folistic"]]["pedon"]==id,] <-
TRUE
        diags[["folistic"]]["top_depth"][diags[["folistic"]]["pedon"]==id,] <-
hz$hzdept[1]
        diags[["folistic"]]["bottom_depth"][diags[["folistic"]]["pedon"]==id,]
<- s[cond1a_db]
        diags[["folistic"]]["thick"][diags[["folistic"]]["pedon"]==id,] <-
s[cond1a_db]
        diags[["folistic"]]["hz"][diags[["folistic"]]["pedon"]==id,] <- TRUE

      }

    } else if(ft==20 & db_wt < 75) {

      cond1a <- FALSE


    } else {

      cond1a <- FALSE

    }

  } else {

    cond1a <- FALSE
```

```
  }



  # Cond 1b: 15cm or more thick
  if(cond1 >= 15 & cond1a==FALSE) {

    cond1b <- TRUE
    diags[["folistic"]]["cond1b"][diags[["folistic"]]["pedon"]==id,] <- TRUE
    diags[["folistic"]]["top_depth"][diags[["folistic"]]["pedon"]==id,] <-
hz$hzdept[1]
    diags[["folistic"]]["bottom_depth"][diags[["folistic"]]["pedon"]==id,] <-
cond1
    diags[["folistic"]]["thick"][diags[["folistic"]]["pedon"]==id,] <- cond1
    diags[["folistic"]]["hz"][diags[["folistic"]]["pedon"]==id,] <- TRUE

  }



  # Cond 2: Ap horizon 25cm organic carbon
  cond2 <- grep("Ap",hz$hzname,fixed=TRUE)

  if(length(cond2)==0) {
    cond2 <- FALSE
    diags[["folistic"]]["cond2a"][diags[["folistic"]]["pedon"]==id,] <- FALSE
    diags[["folistic"]]["cond2b"][diags[["folistic"]]["pedon"]==id,] <- FALSE
    diags[["folistic"]]["cond2c"][diags[["folistic"]]["pedon"]==id,] <- FALSE

    return(diags)
  }

  if(any(hz$hzdepb[cond2] >= 25)) {

    p <- prof_slice(25,hz$hzdept[1],hz)
    t <- p$hzdepb - p$hzdept
    w <- t / 25
    oc <- p$oc
    cl <- p$claytotal
    f_oc <- weighted.mean(oc,w)

    clay_wa <- weighted.mean(cl,w)
    cond2c <- 8 + (clay_wa/7.5)

    if(clay_wa >= 60 & f_oc >= 16) {

      cond2a <- TRUE
      diags[["folistic"]]["cond2a"][diags[["folistic"]]["pedon"]==id,] <- TRUE
      diags[["folistic"]]["top_depth"][diags[["folistic"]]["pedon"]==id,] <-
hz$hzdept[1]
      diags[["folistic"]]["bottom_depth"][diags[["folistic"]]["pedon"]==id,]
<- 25
```

```
      diags[["folistic"]]["thick"][diags[["folistic"]]["pedon"]==id,] <- 25
      diags[["folistic"]]["hz"][diags[["folistic"]]["pedon"]==id,] <- TRUE

    } else if(clay_wa==0 & f_oc >= 8) {

      diags[["folistic"]]["cond2a"][diags[["folistic"]]["pedon"]==id,] <-
FALSE

      cond2b <- TRUE
      diags[["folistic"]]["cond2b"][diags[["folistic"]]["pedon"]==id,] <- TRUE
      diags[["folistic"]]["top_depth"][diags[["folistic"]]["pedon"]==id,] <-
hz$hzdept[1]
      diags[["folistic"]]["bottom_depth"][diags[["folistic"]]["pedon"]==id,]
<- 25
      diags[["folistic"]]["thick"][diags[["folistic"]]["pedon"]==id,] <- 25
      diags[["folistic"]]["hz"][diags[["folistic"]]["pedon"]==id,] <- TRUE

    } else if(clay_wa < 60 & f_oc >= cond2c) {

      diags[["folistic"]]["cond2b"][diags[["folistic"]]["pedon"]==id,] <-
FALSE

      cond2c <- TRUE
      diags[["folistic"]]["cond2c"][diags[["folistic"]]["pedon"]==id,] <- TRUE
      diags[["folistic"]]["top_depth"][diags[["folistic"]]["pedon"]==id,] <-
hz$hzdept[1]
      diags[["folistic"]]["bottom_depth"][diags[["folistic"]]["pedon"]==id,]
<- 25
      diags[["folistic"]]["thick"][diags[["folistic"]]["pedon"]==id,] <- 25
      diags[["folistic"]]["hz"][diags[["folistic"]]["pedon"]==id,] <- TRUE

    } else {

      cond2 <- FALSE
      diags[["folistic"]]["cond2c"][diags[["folistic"]]["pedon"]==id,] <-
FALSE

    }

  } else {

    cond2 <- FALSE
    diags[["folistic"]]["cond2a"][diags[["folistic"]]["pedon"]==id,] <- FALSE
    diags[["folistic"]]["cond2b"][diags[["folistic"]]["pedon"]==id,] <- FALSE
    diags[["folistic"]]["cond2c"][diags[["folistic"]]["pedon"]==id,] <- FALSE

  }


  return(diags)

}
```

```
################################################################################
# TODO:
# - changed bd to oven dry
epi.histic <- function(hzs, diags, id) {


  hz <- dplyr::filter(hzs, pedon_id==id)


  cond0 <- all(hz$cumusat >= 30)

  if(!cond0) {

    diags[["histic"]]["hz"][diags[["histic"]]["pedon"]==id,] <- FALSE
    diags[["histic"]]["cond0"][diags[["histic"]]["pedon"]==id,] <- FALSE
    return(diags)

  }


  # depth of the organic portion
  cond1 <- dplyr::filter(diags[["orgmin"]],pedon==id)[["org_bottom"]]


  if(is.na(cond1)) {

    diags[["histic"]]["hz"][diags[["histic"]]["pedon"]==id,] <- FALSE
    diags[["histic"]]["cond1"][diags[["histic"]]["pedon"]==id,] <- FALSE
    return(diags)

  }



  # Cond 1a: sphagnum and Db
  if(hz$sphgm_vol[1] > 0) {
    idx <- which(hz$sphgm_vol > 0)
    sphgm_idx <- split(idx, cumsum(c(1, diff(idx) != 1)))[[1]]
    v <- hz$sphgm_vol[sphgm_idx]
    t <- hz$hzdepb[sphgm_idx] - hz$hzdept[sphgm_idx]
    ft <- hz$hzdepb[max(sphgm_idx)]
    w <- t/ft
    sphgm_vol <- weighted.mean(v,w)

    if(ft>=20 & ft<=60) {
      s <- seq(21,ft,1)
```

```
      vols <- roll_wa(hz=hz, s=s, prop="sphgm_vol")
      cond1a_sphgm <- max(which(vols >= 75)) # index
      if(length(cond1a_sphgm)==0) {
        cond1a_sphgm <- FALSE

      } else {

        cond1a <- TRUE
        diags[["histic"]]["cond1a"][diags[["histic"]]["pedon"]==id,] <- TRUE
        diags[["histic"]]["top_depth"][diags[["histic"]]["pedon"]==id,] <-
hz$hzdept[1]
        diags[["histic"]]["bottom_depth"][diags[["histic"]]["pedon"]==id,] <-
s[cond1a_sphgm]
        diags[["histic"]]["thick"][diags[["histic"]]["pedon"]==id,] <-
s[cond1a_sphgm]
        diags[["histic"]]["hz"][diags[["histic"]]["pedon"]==id,] <- TRUE

      }

    } else if(ft==20 & sphgm_vol < 75) {

      cond1a_sphgm <- FALSE


    } else {

      cond1a_sphgm <- FALSE

    }


  } else {

    cond1a_sphgm <- FALSE

  }


  # if sphagnum criteria next met, check bulk density criteria
  if(cond1a_sphgm == FALSE & !is.na(hz$dbovendry[1]) & hz$dbovendry[1] < 0.1)
{

    idx <- which(hz$dbovendry >= 0.1)
    db_idx <- split(idx, cumsum(c(1, diff(idx) != 1)))[[1]]

    db <- hz$dbovendry[db_idx]
    t <- hz$hzdepb[db_idx] - hz$hzdept[db_idx]
    ft <- hz$hzdepb[max(db_idx)]
    w <- t/ft
    db_wt <- weighted.mean(db,w)

    if(ft>=20 & ft<=60) {
      s <- seq(21,ft,1)
      dbs <- roll_wa(hz=hz, s=s, prop="dbovendry")
```

```
      cond1a_db <- max(which(dbs >= 75)) # index
      if(length(cond1a_db)==0) {
        cond1a_db <- FALSE

      } else {

        cond1a <- TRUE
        diags[["histic"]]["cond1a"][diags[["histic"]]["pedon"]==id,] <- TRUE
        diags[["histic"]]["top_depth"][diags[["histic"]]["pedon"]==id,] <-
hz$hzdept[1]
        diags[["histic"]]["bottom_depth"][diags[["histic"]]["pedon"]==id,] <-
s[cond1a_db]
        diags[["histic"]]["thick"][diags[["histic"]]["pedon"]==id,] <-
s[cond1a_db]
        diags[["histic"]]["hz"][diags[["histic"]]["pedon"]==id,] <- TRUE

      }

    } else if(ft==20 & db_wt < 75) {

      cond1a_db <- FALSE


    } else {

      cond1a_db <- FALSE

    }

  } else {

    cond1a <- FALSE

  }




  # Cond 1b: 20-40cm  thick
  if(cond1 >= 20 & cond1a==FALSE) {

    if(cond1 > 40) {

      cond1 <- 40

    }

    cond1b <- TRUE
    diags[["histic"]]["cond1b"][diags[["histic"]]["pedon"]==id,] <- cond1b
    diags[["histic"]]["top_depth"][diags[["histic"]]["pedon"]==id,] <-
hz$hzdept[1]
    diags[["histic"]]["bottom_depth"][diags[["histic"]]["pedon"]==id,] <-
cond1
```

```
    diags[["histic"]]["thick"][diags[["histic"]]["pedon"]==id,] <- cond1 -
hz$hzdept[1]
    diags[["histic"]]["hz"][diags[["histic"]]["pedon"]==id,] <- TRUE

    return(diags)

  }



  # Cond 2: Ap horizon 25cm organic carbon
  cond2 <- grep("Ap",hz$hzname,fixed=TRUE)

  if(length(cond2)==0) {
    cond2 <- FALSE
    diags[["histic"]]["cond2a"][diags[["histic"]]["pedon"]==id,] <- FALSE
    diags[["histic"]]["cond2b"][diags[["histic"]]["pedon"]==id,] <- FALSE
    diags[["histic"]]["cond2c"][diags[["histic"]]["pedon"]==id,] <- FALSE

    return(diags)
  }

  if(any(hz$hzdepb[cond2] >= 25)) {

    p <- prof_slice(25,hz$hzdept[1],hz)
    t <- p$hzdepb - p$hzdept
    w <- t / 25
    oc <- p$oc
    cl <- p$claytotal
    f_oc <- weighted.mean(oc,w)

    clay_wa <- weighted.mean(cl,w)
    cond2c <- 8 + (clay_wa/7.5)

    if(clay_wa >= 60 & f_oc >= 16) {

      cond2a <- TRUE
      diags[["histic"]]["cond2a"][diags[["histic"]]["pedon"]==id,] <- TRUE
      diags[["histic"]]["top_depth"][diags[["histic"]]["pedon"]==id,] <-
hz$hzdept[1]
      diags[["histic"]]["bottom_depth"][diags[["histic"]]["pedon"]==id,] <- 25
      diags[["histic"]]["thick"][diags[["histic"]]["pedon"]==id,] <- 25
      diags[["histic"]]["hz"][diags[["histic"]]["pedon"]==id,] <- TRUE

    } else if(clay_wa==0 & f_oc >= 8) {

      diags[["histic"]]["cond2a"][diags[["histic"]]["pedon"]==id,] <- FALSE

      cond2b <- TRUE
      diags[["histic"]]["cond2b"][diags[["histic"]]["pedon"]==id,] <- TRUE
      diags[["histic"]]["top_depth"][diags[["histic"]]["pedon"]==id,] <-
hz$hzdept[1]
      diags[["histic"]]["bottom_depth"][diags[["histic"]]["pedon"]==id,] <- 25
      diags[["histic"]]["thick"][diags[["histic"]]["pedon"]==id,] <- 25
```

```
      diags[["histic"]]["hz"][diags[["histic"]]["pedon"]==id,] <- TRUE

    } else if(clay_wa < 60 & f_oc >= cond2c) {

      diags[["histic"]]["cond2b"][diags[["histic"]]["pedon"]==id,] <- FALSE

      cond2c <- TRUE
      diags[["histic"]]["cond2c"][diags[["histic"]]["pedon"]==id,] <- TRUE
      diags[["histic"]]["top_depth"][diags[["histic"]]["pedon"]==id,] <-
hz$hzdept[1]
      diags[["histic"]]["bottom_depth"][diags[["histic"]]["pedon"]==id,] <- 25
      diags[["histic"]]["thick"][diags[["histic"]]["pedon"]==id,] <- 25
      diags[["histic"]]["hz"][diags[["histic"]]["pedon"]==id,] <- TRUE

    } else {

      cond2 <- FALSE
      diags[["histic"]]["cond2c"][diags[["histic"]]["pedon"]==id,] <- FALSE

    }

  } else {

    cond2 <- FALSE

  }


  return(diags)




}









################################################################################
# TODO:
epi.melanic <- function(hzs, diags, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)

  # initial indices of hz for hzs with andic properties
  cond2a <- which(hz$andic)

  if(length(cond2a)!=0) {
```

```
    diags[["melanic"]]["cond2a"][diags[["melanic"]]["pedon"]==id,] <- TRUE
  } else {
    diags[["melanic"]]["cond2a"][diags[["melanic"]]["pedon"]==id,] <- FALSE
    diags[["melanic"]]["hz"][diags[["melanic"]]["pedon"]==id,] <- FALSE
    return(diags)
  }



  cond2b <- cond2a[cond2a %in% which( (hz$colorvalue_w <= 2.5) &
(hz$colorchroma_w <= 2) )]

  if(length(cond2b)!=0) {
    diags[["melanic"]]["cond2b"][diags[["melanic"]]["pedon"]==id,] <- TRUE
  } else {
    diags[["melanic"]]["cond2b"][diags[["melanic"]]["pedon"]==id,] <- FALSE
    diags[["melanic"]]["hz"][diags[["melanic"]]["pedon"]==id,] <- FALSE
    return(diags)
  }



  cond2c <- cond2b[cond2b %in% which(hz$melanic_idx <= 1.70)]

  if(length(cond2c)!=0) {
    diags[["melanic"]]["cond2c"][diags[["melanic"]]["pedon"]==id,] <- TRUE
  } else {
    diags[["melanic"]]["cond2c"][diags[["melanic"]]["pedon"]==id,] <- FALSE
    diags[["melanic"]]["hz"][diags[["melanic"]]["pedon"]==id,] <- FALSE
    return(diags)
  }



  cond2d_oc4 <- cond2c[cond2c %in% which(hz$oc >= 4)]

  if(length(cond2d_oc4)!=0) {
    diags[["melanic"]]["cond2d_oc4"][diags[["melanic"]]["pedon"]==id,] <- TRUE
  } else {
    diags[["melanic"]]["cond2d_oc4"][diags[["melanic"]]["pedon"]==id,] <-
FALSE
    diags[["melanic"]]["hz"][diags[["melanic"]]["pedon"]==id,] <- FALSE
    return(diags)
  }




  m <- min(cond2d_oc4)
  org_andic <- which(hz$desgnmaster == "O" & hz$andic)
  if(length(org_andic)!=0) {
    org_andic_idx <- min(org_andic)
    org_andic_top <- hz$hzdept[org_andic_idx]
```

```
    if(org_andic_top < dplyr::filter(diags[["orgmin"]],pedon==id)
[["min_top"]]) {
      shallower <- org_andic_top
    } else {
      shallower <- dplyr::filter(diags[["orgmin"]],pedon==id)[["min_top"]]
    }

  } else {
    shallower <- dplyr::filter(diags[["orgmin"]],pedon==id)[["min_top"]]
  }




  if(hz$hzdept[m] >= shallower & hz$hzdept[m] <= shallower+30) {
    cond1 <- TRUE
    diags[["melanic"]]["cond1"][diags[["melanic"]]["pedon"]==id,] <- TRUE
  } else {
    cond1 <- FALSE
    diags[["melanic"]]["cond1"][diags[["melanic"]]["pedon"]==id,] <- FALSE
    diags[["melanic"]]["hz"][diags[["melanic"]]["pedon"]==id,] <- FALSE
    return(diags)
  }


  ## what is this doing?
  # if(cond1 & cond2d_oc4[2] == m+1) {
  if(cond1 & length(cond2d_oc4)!=0) {

    s <- get_seq(cond2d_oc4)
    if(sum(hz$hzthk[s]) >= 30) {
      cond2 <- s
      diags[["melanic"]]["cond2"][diags[["melanic"]]["pedon"]==id,] <- TRUE
    } else {
      diags[["melanic"]]["cond2"][diags[["melanic"]]["pedon"]==id,] <- FALSE
      diags[["melanic"]]["hz"][diags[["melanic"]]["pedon"]==id,] <- FALSE
      return(diags)
    }

  # } else if(cond1) {
  #
  #   if(hz$hzthk[m] >= 30) {
  #     cond2 <- m
  #     diags[["melanic"]]["cond2"][diags[["melanic"]]["pedon"]==id,] <- TRUE
  #   } else {
  #     diags[["melanic"]]["cond2"][diags[["melanic"]]["pedon"]==id,] <- FALSE
  #     diags[["melanic"]]["hz"][diags[["melanic"]]["pedon"]==id,] <- FALSE
  #     return(diags)
  #   }

  } else {
    cond2 <- vector()
    diags[["melanic"]]["cond2"][diags[["melanic"]]["pedon"]==id,] <- NA
```

```
  }



  if(length(cond2)!=0) {
    cond2d_oc6wa <- weighted.mean(hz$oc[cond2],hz$hzthk[cond2]/
sum(hz$hzthk[cond2]))

    if(cond2d_oc6wa >= 6) {
      diags[["melanic"]]["cond2d_oc6wa"][diags[["melanic"]]["pedon"]==id,] <-
TRUE
      diags[["melanic"]][""][diags[["melanic"]]["pedon"]==id,] <- TRUE
      diags[["melanic"]]["top_depth"][diags[["melanic"]]["pedon"]==id,] <-
hz$hzdept[min(cond2)]
      diags[["melanic"]]["bottom_depth"][diags[["melanic"]]["pedon"]==id,] <-
hz$hzdept[max(cond2)]
      diags[["melanic"]]["thick"][diags[["melanic"]]["pedon"]==id,] <-
hz$hzdept[max(cond2)] - hz$hzdept[min(cond2)]

    } else {
      diags[["melanic"]]["cond2d_oc6wa"][diags[["melanic"]]["pedon"]==id,] <-
FALSE
      diags[["melanic"]]["hz"][diags[["melanic"]]["pedon"]==id,] <- FALSE
      return(diags)
    }

  } else {
    cond2d_oc6wa <- NA
    diags[["melanic"]]["cond2d_oc6wa"][diags[["melanic"]]["pedon"]==id,] <- NA

  }



  return(diags)


}









################################################################################
# TODO:
# - removed cambic checks
# - secondary carbonates not checked
# - condition 7 not checked
epi.mollic <- function(hzs, diags, id) {
```

```
  hz <- dplyr::filter(hzs, pedon_id==id)


  ## prevent both histic and mollic
  if(diags[["histic"]]["hz"][diags[["histic"]]["pedon"]==id,]) {

    diags[["mollic"]]["hz"][diags[["mollic"]]["pedon"]==id,] <- FALSE

    return(diags)

  }


  ## Cond 1

  cond1a <- which(
    tolower(hz$structtype) %in% c("granular","platy") |
      (tolower(hz$structtype) %in% c("lenticular","angular blocky","subangular
blocky") & hz$structsize != "very coarse") |
      (tolower(hz$structtype) %in% c("prismatic","columnar","wedge") &
hz$structsize %in% c("very fine","fine","medium"))
  )


  cond1a <- split(cond1a, cumsum(c(1, diff(cond1a) != 1)))[[1]]

  cond1b <- which(tolower(hz$rrc) %in% c("moderately hard","slightly
hard","soft","loose",
                                         "extremely weak","very
weak","weak","moderate",
                                         "mh","sh","s","l","ew","vw","w","m"))


  if(length(cond1a)==0) {
    diags[["mollic"]]["cond1a"][diags[["mollic"]]["pedon"]==id,] <- FALSE
  } else {
    diags[["mollic"]]["cond1a"][diags[["mollic"]]["pedon"]==id,] <- TRUE
  }

  if(length(cond1b)==0) {
    diags[["mollic"]]["cond1b"][diags[["mollic"]]["pedon"]==id,] <- FALSE
  } else {
    diags[["mollic"]]["cond1b"][diags[["mollic"]]["pedon"]==id,] <- TRUE
  }

  if(length(cond1a)==0 & length(cond1b)==0) {
    diags[["mollic"]]["cond1a"][diags[["mollic"]]["pedon"]==id,] <- FALSE
    diags[["mollic"]]["cond1b"][diags[["mollic"]]["pedon"]==id,] <- FALSE
    diags[["mollic"]]["hz"][diags[["mollic"]]["pedon"]==id,] <- FALSE
    return(diags)
  }
```

```
  cond1 <- list(cond1a,cond1b)
  cond1_hz1 <- unlist(lapply(cond1, function(x) { 1 %in% x }))
  cond1 <- cond1[cond1_hz1]

  if(length(cond1)==2){
    cond1 <- cond1[which.max(lengths(cond1))]
  }
  cond1 <- unlist(cond1)









  ## Cond 2

  cond2 <- cond1[cond1 %in% which(hz$fragtotvol < 50)]

  if(length(cond2)==0) {
    diags[["mollic"]]["cond2"][diags[["mollic"]]["pedon"]==id,] <- FALSE
    diags[["mollic"]]["hz"][diags[["mollic"]]["pedon"]==id,] <- FALSE
    return(diags)
  } else {
    diags[["mollic"]]["cond2"][diags[["mollic"]]["pedon"]==id,] <- TRUE
  }









  ## Cond 3: 3a (1 AND 2) OR 3b OR 3c

  cond3a1 <- cond2[cond2 %in% which(hz$colorvalue_w <= 3 & hz$colorvalue_d <=
5)]
  cond3a2 <- cond3a1[cond3a1 %in% which(hz$colorchroma_w <= 3)]

  if(length(cond3a1)==0) {
    diags[["mollic"]]["cond3a1"][diags[["mollic"]]["pedon"]==id,] <- FALSE
    diags[["mollic"]]["hz"][diags[["mollic"]]["pedon"]==id,] <- FALSE
    return(diags)
  } else {
    diags[["mollic"]]["cond3a1"][diags[["mollic"]]["pedon"]==id,] <- TRUE
  }

  if(length(cond3a2)==0) {
    diags[["mollic"]]["cond3a2"][diags[["mollic"]]["pedon"]==id,] <- FALSE
    diags[["mollic"]]["hz"][diags[["mollic"]]["pedon"]==id,] <- FALSE
    return(diags)
  } else {
```

```
    diags[["mollic"]]["cond3a2"][diags[["mollic"]]["pedon"]==id,] <- TRUE
  }



  cond3b <- cond2[cond2 %in% which((hz$claysizedcarb[cond2] >= 15 &
hz$claysizedcarb[cond2] <= 40) &
                 (hz$colorvalue_w[cond2] <= 3 & hz$colorchroma_w[cond2] <=
3))]

  if(length(cond3b)==0) {
    diags[["mollic"]]["cond3b"][diags[["mollic"]]["pedon"]==id,] <- FALSE
    diags[["mollic"]]["hz"][diags[["mollic"]]["pedon"]==id,] <- FALSE

  } else {
    diags[["mollic"]]["cond3b"][diags[["mollic"]]["pedon"]==id,] <- TRUE
  }



  cond3c <- cond2[cond2 %in% which((hz$claysizedcarb[cond2] >= 40) &
                 (hz$colorvalue_w[cond2] <= 5))]

  if(length(cond3c)==0) {
    diags[["mollic"]]["cond3c"][diags[["mollic"]]["pedon"]==id,] <- FALSE
    diags[["mollic"]]["hz"][diags[["mollic"]]["pedon"]==id,] <- FALSE

  } else {
    diags[["mollic"]]["cond3c"][diags[["mollic"]]["pedon"]==id,] <- TRUE
  }


  cond3 <- list(cond3a2,cond3b,cond3c)
  cond3_hz1 <- unlist(lapply(cond3, function(x) { 1 %in% x }))
  cond3 <- cond3[cond3_hz1]
  cond3 <- cond3[which.max(lapply(cond3,length))]
  cond3 <- unlist(cond3)


  if(length(cond3)==0) {
    diags[["mollic"]]["cond3"][diags[["mollic"]]["pedon"]==id,] <- FALSE
    diags[["mollic"]]["hz"][diags[["mollic"]]["pedon"]==id,] <- FALSE
    return(diags)
  } else {
    diags[["mollic"]]["cond3"][diags[["mollic"]]["pedon"]==id,] <- TRUE
  }



  ## Cond 4

  cond4 <- cond3[cond3 %in% which(hz$bs82 >= 50)]
```

```
  if(length(cond4)==0) {
    diags[["mollic"]]["cond4"][diags[["mollic"]]["pedon"]==id,] <- FALSE
    diags[["mollic"]]["hz"][diags[["mollic"]]["pedon"]==id,] <- FALSE
    return(diags)
  } else {
    diags[["mollic"]]["cond4"][diags[["mollic"]]["pedon"]==id,] <- TRUE
  }




  ## Cond 5: 5a OR 5b OR 5c


  ## Cond 5a

  cond5a_colorval <-cond4[cond4 %in% which(hz$colorvalue_w %in% c(4,5))]
  cond5a_colorval_idx <- split(cond5a_colorval, cumsum(c(1,
diff(cond5a_colorval) != 1)))[[1]]

  cond5a <- cond5a_colorval_idx[cond5a_colorval_idx %in% which(hz$oc >= 2.5)]

  if(length(cond5a)==0) {
    diags[["mollic"]]["cond5a"][diags[["mollic"]]["pedon"]==id,] <- FALSE
  } else {
    diags[["mollic"]]["cond5a"][diags[["mollic"]]["pedon"]==id,] <- TRUE
  }



  ## Cond 5b

  if(any("C" %in% hz$desgnmaster)) {

    c_idx <- which.min(hz$desgnmaster == "C")

    c_value_w <- hz$colorvalue_w[c_idx]
    c_value_d <- hz$colorvalue_d[c_idx]

    c_chroma_w <- hz$colorchroma_w[c_idx]
    c_chroma_d <- hz$colorchroma_d[c_idx]

    c_oc <- hz$oc[c_idx]

    cond5b_val_w <- cond4[cond4 %in% which(hz$colorvalue_w < c_value_w-1)]

    cond5b_chroma_w <- cond4[cond4 %in% which(hz$colorchroma_w <
c_chroma_w-2)]
    cond5b_chroma_d <- cond4[cond4 %in% which(hz$colorchroma_d <
c_chroma_d-2)]

    cond5b_chroma <- list(cond5b_chroma_w,cond5b_chroma_d)
```

```
    cond5b_chroma_hz1 <- unlist(lapply(cond5b_chroma, function(x) { 1 %in% x
}))
    cond5b_chroma <- cond5b_chroma[cond5b_chroma_hz1]
    cond5b_chroma <- cond5b_chroma[which.max(lapply(cond5b_chroma,length))]


    cond5b_idx <- list(cond5b_val_w,cond5b_chroma)
    cond5b_hz1 <- unlist(lapply(cond5b_idx, function(x) { 1 %in% x }))
    cond5b_idx <- cond5b_idx[cond5b_hz1]
    cond5b_idx <- cond5b_idx[which.max(lapply(cond5b_idx,length))]


    cond5b <- cond5b_idx[cond5b_idx %in% which(hz$oc > 1.006*c_oc)]

    if(length(cond5b)==0) {
      diags[["mollic"]]["cond5b"][diags[["mollic"]]["pedon"]==id,] <- FALSE
    } else {
      diags[["mollic"]]["cond5b"][diags[["mollic"]]["pedon"]==id,] <- TRUE
    }


  } else {
    cond5b <- vector()
    diags[["mollic"]]["cond5b"][diags[["mollic"]]["pedon"]==id,] <- FALSE
  }



  ## Cond 5c

  if(diags[["mollic"]]["cond5a"][diags[["mollic"]]["pedon"]==id,] == FALSE &
     diags[["mollic"]]["cond5b"][diags[["mollic"]]["pedon"]==id,] == FALSE) {

    cond5c <- cond4[cond4 %in% which(hz$oc >= 0.6)]

    if(length(cond5c)==0) {
      diags[["mollic"]]["cond5c"][diags[["mollic"]]["pedon"]==id,] <- FALSE
      diags[["mollic"]]["hz"][diags[["mollic"]]["pedon"]==id,] <- FALSE
    } else {
      diags[["mollic"]]["cond5c"][diags[["mollic"]]["pedon"]==id,] <- TRUE
    }

  } else {
    diags[["mollic"]]["cond5c"][diags[["mollic"]]["pedon"]==id,] <- NA
  }


  cond5 <- list(cond5a,cond5b,cond5c)
  cond5_hz1 <- unlist(lapply(cond5, function(x) { 1 %in% x }))
  cond5 <- cond5[cond5_hz1]
  cond5 <- cond5[which.max(lapply(cond5,length))]
  cond5 <- unlist(cond5)

  if(length(cond5)==0){
```

```
    diags[["mollic"]]["hz"][diags[["mollic"]]["pedon"]==id,] <- FALSE
    return(diags)

  }

  diags[["mollic"]]["cond5"][diags[["mollic"]]["pedon"]==id,] <- TRUE


  ## Cond 6: Minimum thickness
  ## Assumes all hz meeting criteria up to here are within epipedon

  ## Cond 6a: 25cm if 6a1, 6a2, OR 6a3



  ## Cond 6a

  if(sum(hz$hzthk[cond5]) >= 25) {


    ## Cond 6a1
    cond6a1 <- cond5[cond5 %in% which(tolower(hz$texture) %in% c("loamy fine
sand","loamy sand","loamy coarse sand",
                                                        "very fine sand","fine
sand","sand","coarse sand",
                                                        "lfs","ls","lcos","vfs","fs","s","c

    if(sum(hz$hzthk[cond6a1]) >= 25) {

      # if(sum(hz$hzthk[cond6a1]) > 25) {
      #   diags[["mollic"]]["bottom_depth"][diags[["mollic"]]["pedon"]==id,]
<- 25
      #   diags[["mollic"]]["thick"][diags[["mollic"]]["pedon"]==id,] <- 25
      # } else {
      #   diags[["mollic"]]["bottom_depth"][diags[["mollic"]]["pedon"]==id,]
<- sum(hz$hzthk[cond6a1])
      #   diags[["mollic"]]["thick"][diags[["mollic"]]["pedon"]==id,] <-
sum(hz$hzthk[cond6a1])
      # }

      diags[["mollic"]]["cond6a1"][diags[["mollic"]]["pedon"]==id,] <- TRUE
      diags[["mollic"]]["hz"][diags[["mollic"]]["pedon"]==id,] <- TRUE
      diags[["mollic"]]["top_depth"][diags[["mollic"]]["pedon"]==id,] <-
hz$hzdept[1]
      diags[["mollic"]]["bottom_depth"][diags[["mollic"]]["pedon"]==id,] <-
sum(hz$hzthk[cond6a1])
      diags[["mollic"]]["thick"][diags[["mollic"]]["pedon"]==id,] <-
sum(hz$hzthk[cond6a1])

      return(diags)

    } else {
```

```
      diags[["mollic"]]["cond6a1"][diags[["mollic"]]["pedon"]==id,] <- FALSE

      ## cond6a2
      diag_sub <- c("agric", "albic", "anhydric", "argillic", "calcic",
                          "duripan", "fragipan", "glossic", "gypsic",
                          "kandic", "natric", "ortstein", "oxic",
"petrocalcic",
                          "petrogypsic", "placic", "salic", "sombric",
"spodic")

      if(!any(as.logical(lapply(diags[diag_sub],function(x){x[["hz"]]
[which(x[["pedon"]]==id)]})))) {

        # if(sum(hz$hzthk[cond5]) > 25) {
        #   diags[["mollic"]]["bottom_depth"][diags[["mollic"]]["pedon"]==id,]
<- 25
        #   diags[["mollic"]]["thick"][diags[["mollic"]]["pedon"]==id,] <- 25
        # } else {
        #   diags[["mollic"]]["bottom_depth"][diags[["mollic"]]["pedon"]==id,]
<- sum(hz$hzthk[cond5])
        #   diags[["mollic"]]["thick"][diags[["mollic"]]["pedon"]==id,] <-
sum(hz$hzthk[cond5])
        # }

        ## Not checked: carbon content decreases irregularly with depth
        diags[["mollic"]]["cond6a2"][diags[["mollic"]]["pedon"]==id,] <- TRUE
        diags[["mollic"]]["hz"][diags[["mollic"]]["pedon"]==id,] <- TRUE
        diags[["mollic"]]["top_depth"][diags[["mollic"]]["pedon"]==id,] <-
hz$hzdept[1]
        diags[["mollic"]]["bottom_depth"][diags[["mollic"]]["pedon"]==id,] <-
sum(hz$hzthk[cond5])
        diags[["mollic"]]["thick"][diags[["mollic"]]["pedon"]==id,] <-
sum(hz$hzthk[cond5])


        return(diags)

      } else {



        diags[["mollic"]]["cond6a2"][diags[["mollic"]]["pedon"]==id,] <- FALSE

        ## cond 6a3

        ## Cond6a3a

        ## Not checked: secondary carbonates
        cond6a3a_hz <- c("calcic","petrocalcic","duripan","fragipan")
        if(any(as.logical(lapply(diags[cond6a3a_hz],function(x){x[["hz"]]
[which(x[["pedon"]]==id)]})))) {
```

```
          cond6a3a_hz_idx <-
which(as.logical(lapply(diags[cond6a3a_hz],function(x){x[["hz"]]
[which(x[["pedon"]]==id)]})))
          cond6a3a <- any(diags[[cond6a3a_hz[cond6a3a_hz_idx]]]["top_depth"]
[diags[[cond6a3a_hz[cond6a3a_hz_idx]]]["pedon"]==id,] <= 75+diags[["orgmin"]]
["min_top"][diags[["orgmin"]]["pedon"]==id,])

          if(cond6a3a) {

            cond6a3a <- TRUE
            diags[["mollic"]]["cond6a3a"][diags[["mollic"]]["pedon"]==id,] <-
TRUE

          } else {

            cond6a3a <- FALSE
            diags[["mollic"]]["cond6a3a"][diags[["mollic"]]["pedon"]==id,] <-
FALSE

          }


        } else {

          cond6a3a <- FALSE
          diags[["mollic"]]["cond6a3a"][diags[["mollic"]]["pedon"]==id,] <-
FALSE

        }


        ## Cond6a3b

        cond6a3b_hz <- c("argillic","natric","oxic","spodic")
        if(any(as.logical(lapply(diags[cond6a3b_hz],function(x){x[["hz"]]
[which(x[["pedon"]]==id)]})))) {

          cond6a3b_hz_idx <-
which(as.logical(lapply(diags[cond6a3b_hz],function(x){x[["hz"]]
[which(x[["pedon"]]==id)]})))
          cond6a3b <- max(diags[[cond6a3b_hz[cond6a3b_hz_idx]]]
["bottom_depth"][diags[[cond6a3b_hz[cond6a3b_hz_idx]]]["pedon"]==id,]) <=
75+diags[["orgmin"]]["min_top"][diags[["orgmin"]]["pedon"]==id,]


          if(cond6a3b) {

            cond6a3b <- TRUE
            diags[["mollic"]]["cond6a3b"][diags[["mollic"]]["pedon"]==id,] <-
TRUE

          } else {

            cond6a3b <- FALSE
```

```
            diags[["mollic"]]["cond6a3b"][diags[["mollic"]]["pedon"]==id,] <-
FALSE

          }

        } else {

          cond6a3b <- FALSE
          diags[["mollic"]]["cond6a3b"][diags[["mollic"]]["pedon"]==id,] <-
FALSE

        }


        ## final check for Cond6a3

        if(cond6a3a | cond6a3b) {

          # if(sum(hz$hzthk[cond5]) > 25) {
          #   diags[["mollic"]]["bottom_depth"][diags[["mollic"]]
["pedon"]==id,] <- 25
          #   diags[["mollic"]]["thick"][diags[["mollic"]]["pedon"]==id,] <-
25
          # } else {
          #   diags[["mollic"]]["bottom_depth"][diags[["mollic"]]
["pedon"]==id,] <- sum(hz$hzthk[cond5])
          #   diags[["mollic"]]["thick"][diags[["mollic"]]["pedon"]==id,] <-
sum(hz$hzthk[cond5])
          # }

          diags[["mollic"]]["cond6a3"][diags[["mollic"]]["pedon"]==id,] <-
TRUE
          diags[["mollic"]]["hz"][diags[["mollic"]]["pedon"]==id,] <- TRUE
          diags[["mollic"]]["top_depth"][diags[["mollic"]]["pedon"]==id,] <-
hz$hzdept[1]
          diags[["mollic"]]["bottom_depth"][diags[["mollic"]]["pedon"]==id,]
<- sum(hz$hzthk[cond5])
          diags[["mollic"]]["thick"][diags[["mollic"]]["pedon"]==id,] <-
sum(hz$hzthk[cond5])

          return(diags)

        } else {

          diags[["mollic"]]["cond6a3"][diags[["mollic"]]["pedon"]==id,] <-
FALSE

        }



      }

    }
```

```
  } else {

    diags[["mollic"]]["cond6a"][diags[["mollic"]]["pedon"]==id,] <- FALSE


  }




  ## Cond 6b: 10cm if 6b

  ## Cond 6b

  if(sum(hz$hzthk[cond5]) >= 10) {

    cond6b <- cond5[cond5 %in% which(!tolower(hz$texture) %in% c("loamy fine
sand","loamy sand","loamy coarse sand",
                                                          "very fine sand","fine
sand","sand","coarse sand",
                                                          "lfs","ls","lcos","vfs","fs","s","c

    cond6b_bottom <- max(hz$hzdepb[cond6b])
    cond6b_hz <- c("densic","lithic","paralithic","petrocalcic","duripan")
    cond6b_hz_idx <- which(as.logical(lapply(diags[cond6b_hz],function(x)
{x[["hz"]][which(x[["pedon"]]==id)]})))
    if(length(cond6b_hz_idx)==0){
      cond6b_hz_tops <- cond6b_bottom+1
    } else {
      cond6b_hz_tops <- diags[[cond6b_hz[cond6b_hz_idx]]]["top_depth"]
[diags[[cond6b_hz[cond6b_hz_idx]]]["pedon"]==id,]
    }



    if(any(cond6b_hz_tops == cond6b_bottom)) {

      diags[["mollic"]]["cond6b"][diags[["mollic"]]["pedon"]==id,] <- TRUE
      diags[["mollic"]]["hz"][diags[["mollic"]]["pedon"]==id,] <- TRUE
      diags[["mollic"]]["top_depth"][diags[["mollic"]]["pedon"]==id,] <-
hz$hzdept[1]
      diags[["mollic"]]["bottom_depth"][diags[["mollic"]]["pedon"]==id,] <-
cond6b_bottom
      diags[["mollic"]]["thick"][diags[["mollic"]]["pedon"]==id,] <-
cond6b_bottom

      return(diags)

    } else {

      diags[["mollic"]]["cond6b"][diags[["mollic"]]["pedon"]==id,] <- FALSE
```

```
    }

  }



  ## Cond 6c: 18-25cm if 6c1 and/or 6c2


  cond6c1_hz <- c("calcic","petrocalcic","duripan","fragipan")
  cond6c2_hz <- c("argillic","cambic","natric","oxic","spodic")

  min_top <- dplyr::filter(diags[["orgmin"]],pedon==id)[["min_top"]]

  ## Cond 6c1
  cond6c1_hz_idx <- which(as.logical(lapply(diags[cond6c1_hz],function(x)
{x[["hz"]][which(x[["pedon"]]==id)]})))
  if(length(cond6c1_hz_idx)!=0 | sum(hz$hzthk[cond5]) > 18) {

     if(length(cond6c1_hz_idx)==0) {
       cond6c1_thick <- TRUE
     } else {
       cond6c1_thick <- min_top - min(diags[[cond6c1_hz[cond6c1_hz_idx]]]
["top_depth"][diags[[cond6c1_hz[cond6c1_hz_idx]]]["pedon"]==id,])
     }


     if(sum(hz$hzthk[cond5]) >= 0.33*cond6c1_thick) {

       # if(sum(hz$hzthk[cond5]) > 25) {
       #   diags[["mollic"]]["bottom_depth"][diags[["mollic"]]["pedon"]==id,]
<- 25
       #   diags[["mollic"]]["thick"][diags[["mollic"]]["pedon"]==id,] <- 25
       # } else {
       #   diags[["mollic"]]["bottom_depth"][diags[["mollic"]]["pedon"]==id,]
<- sum(hz$hzthk[cond5])
       #   diags[["mollic"]]["thick"][diags[["mollic"]]["pedon"]==id,] <-
sum(hz$hzthk[cond5])
       # }

       diags[["mollic"]]["cond6c1"][diags[["mollic"]]["pedon"]==id,] <- TRUE
       diags[["mollic"]]["hz"][diags[["mollic"]]["pedon"]==id,] <- TRUE
       diags[["mollic"]]["top_depth"][diags[["mollic"]]["pedon"]==id,] <-
hz$hzdept[1]
       diags[["mollic"]]["bottom_depth"][diags[["mollic"]]["pedon"]==id,] <-
sum(hz$hzthk[cond5])
       diags[["mollic"]]["thick"][diags[["mollic"]]["pedon"]==id,] <-
sum(hz$hzthk[cond5])
```

```
      return(diags)

    }

  } else {

    diags[["mollic"]]["cond6c1"][diags[["mollic"]]["pedon"]==id,] <- FALSE

  }


  cond6c2_hz_idx <- which(as.logical(lapply(diags[cond6c2_hz],function(x)
{x[["hz"]][which(x[["pedon"]]==id)]})))
  if(length(cond6c2_hz_idx)!=0 | sum(hz$hzthk[cond5]) >= 18) {


    if(length(cond6c2_hz_idx)==0) {
      cond6c2_thick <- TRUE
    } else {
      cond6c2_thick <- min_top - min(diags[[cond6c2_hz[cond6c2_hz_idx]]]
["top_depth"][diags[[cond6c2_hz[cond6c2_hz_idx]]]["pedon"]==id,])
    }


    if(sum(hz$hzthk[cond5]) >= 0.33*cond6c2_thick) {

      # if(sum(hz$hzthk[cond5]) > 25) {
      #   diags[["mollic"]]["bottom_depth"][diags[["mollic"]]["pedon"]==id,]
<- 25
      #   diags[["mollic"]]["thick"][diags[["mollic"]]["pedon"]==id,] <- 25
      # } else {
      #   diags[["mollic"]]["bottom_depth"][diags[["mollic"]]["pedon"]==id,]
<- sum(hz$hzthk[cond5])
      #   diags[["mollic"]]["thick"][diags[["mollic"]]["pedon"]==id,] <-
sum(hz$hzthk[cond5])
      # }

      diags[["mollic"]]["cond6c2"][diags[["mollic"]]["pedon"]==id,] <- TRUE
      diags[["mollic"]]["hz"][diags[["mollic"]]["pedon"]==id,] <- TRUE
      diags[["mollic"]]["top_depth"][diags[["mollic"]]["pedon"]==id,] <-
hz$hzdept[1]
      diags[["mollic"]]["bottom_depth"][diags[["mollic"]]["pedon"]==id,] <-
sum(hz$hzthk[cond5])
      diags[["mollic"]]["thick"][diags[["mollic"]]["pedon"]==id,] <-
sum(hz$hzthk[cond5])

      return(diags)

    }

  } else {

    diags[["mollic"]]["cond6c2"][diags[["mollic"]]["pedon"]==id,] <- FALSE
```

```
  }



  ## Cond 6d: 18cm if 7 AND 8

  ## CONDITION 7 NOT CHECKED

  cond8 <- cond5[which(hz$n < 0.7) %in% cond5]
  if(length(cond8)!=0 & sum(hz$hzthk[cond8]) >= 18) {

    # if(sum(hz$hzthk[cond8]) > 18) {
    #   diags[["mollic"]]["bottom_depth"][diags[["mollic"]]["pedon"]==id,] <-
18
    #   diags[["mollic"]]["thick"][diags[["mollic"]]["pedon"]==id,] <- 18
    # } else {
    #   diags[["mollic"]]["bottom_depth"][diags[["mollic"]]["pedon"]==id,] <-
sum(hz$hzthk[cond8])
    #   diags[["mollic"]]["thick"][diags[["mollic"]]["pedon"]==id,] <-
sum(hz$hzthk[cond8])
    # }

    diags[["mollic"]]["cond8"][diags[["mollic"]]["pedon"]==id,] <- TRUE
    diags[["mollic"]]["hz"][diags[["mollic"]]["pedon"]==id,] <- TRUE
    diags[["mollic"]]["top_depth"][diags[["mollic"]]["pedon"]==id,] <-
hz$hzdept[1]
    diags[["mollic"]]["bottom_depth"][diags[["mollic"]]["pedon"]==id,] <-
sum(hz$hzthk[cond8])
    diags[["mollic"]]["thick"][diags[["mollic"]]["pedon"]==id,] <-
sum(hz$hzthk[cond8])


    return(diags)

  } else {

    diags[["mollic"]]["cond8"][diags[["mollic"]]["pedon"]==id,] <- FALSE
    diags[["mollic"]]["hz"][diags[["mollic"]]["pedon"]==id,] <- FALSE

    return(diags)
  }



}
```

```
################################################################################
# TODO:
# - removed cambic checks
# - secondary carbonates not checked
epi.umbric <- function(hzs, diags, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)


  ## Check if not organic
  o_thk <- sum(hz$hzthk[hz$desgnmaster == "O"])

  if(o_thk > 20) {

    diags[["umbric"]]["hz"][diags[["umbric"]]["pedon"]==id,] <- FALSE

    return(diags)

  }


  ## Cond 1

  cond1a <- which(
    tolower(hz$structtype) %in% c("granular","platy") |
      (tolower(hz$structtype) %in% c("lenticular","angular blocky","subangular
blocky") & hz$structsize != "very coarse") |
      (tolower(hz$structtype) %in% c("prismatic","columnar","wedge") &
hz$structsize %in% c("very fine","fine","medium"))
  )


  cond1a <- split(cond1a, cumsum(c(1, diff(cond1a) != 1)))[[1]]

  cond1b <- which(tolower(hz$rrc) %in% c("moderately hard","slightly
hard","soft","loose",
                                         "extremely weak","very
weak","weak","moderate",
                                         "mh","sh","s","l","ew","vw","w","m"))


  if(length(cond1a)==0) {
    diags[["umbric"]]["cond1a"][diags[["umbric"]]["pedon"]==id,] <- FALSE
  } else {
    diags[["umbric"]]["cond1a"][diags[["umbric"]]["pedon"]==id,] <- TRUE
  }

  if(length(cond1b)==0) {
    diags[["umbric"]]["cond1b"][diags[["umbric"]]["pedon"]==id,] <- FALSE
  } else {
```

```
    diags[["umbric"]]["cond1b"][diags[["umbric"]]["pedon"]==id,] <- TRUE
  }

  if(length(cond1a)==0 & length(cond1b)==0) {
    diags[["umbric"]]["cond1a"][diags[["umbric"]]["pedon"]==id,] <- FALSE
    diags[["umbric"]]["cond1b"][diags[["umbric"]]["pedon"]==id,] <- FALSE
    diags[["umbric"]]["hz"][diags[["umbric"]]["pedon"]==id,] <- FALSE
    return(diags)
  }


  cond1 <- list(cond1a,cond1b)
  cond1_hz1 <- unlist(lapply(cond1, function(x) { 1 %in% x }))
  cond1 <- cond1[cond1_hz1]

  if(length(cond1)==2){
    cond1 <- cond1[which.max(lengths(cond1))]
  }
  cond1 <- unlist(cond1)









  ## Cond 2

  cond2 <- cond1[cond1 %in% which(hz$fragtotvol < 50)]

  if(length(cond2)==0) {
    diags[["umbric"]]["cond2"][diags[["umbric"]]["pedon"]==id,] <- FALSE
    diags[["umbric"]]["hz"][diags[["umbric"]]["pedon"]==id,] <- FALSE
    return(diags)
  } else {
    diags[["umbric"]]["cond2"][diags[["umbric"]]["pedon"]==id,] <- TRUE
  }









  ## Cond 3: both 3a AND 3b

  cond3a <- cond2[cond2 %in% which(hz$colorvalue_w <= 3 & hz$colorvalue_d <=
5)]
  cond3b <- cond3a[cond3a %in% which(hz$colorchroma_w <= 3)]

  if(length(cond3b)==0) {
```

```
    diags[["umbric"]]["cond3"][diags[["umbric"]]["pedon"]==id,] <- FALSE
    diags[["umbric"]]["hz"][diags[["umbric"]]["pedon"]==id,] <- FALSE

    return(diags)

  } else {

    diags[["umbric"]]["cond3"][diags[["umbric"]]["pedon"]==id,] <- TRUE
    cond3 <- cond3b

  }




  ## Cond 4

  cond4 <- cond3[cond3 %in% which(hz$bs82 <= 50)]

  if(length(cond4)==0) {
    diags[["umbric"]]["cond4"][diags[["umbric"]]["pedon"]==id,] <- FALSE
    diags[["umbric"]]["hz"][diags[["umbric"]]["pedon"]==id,] <- FALSE
    return(diags)
  } else {
    diags[["umbric"]]["cond4"][diags[["umbric"]]["pedon"]==id,] <- TRUE
  }




  ## Cond 5: 5a OR 5b



  ## Cond 5a

  if(any("C" %in% hz$desgnmaster)) {

    c_idx <- which.min(hz$desgnmaster == "C")

    c_value_w <- hz$colorvalue_w[c_idx]
    c_value_d <- hz$colorvalue_d[c_idx]

    c_chroma_w <- hz$colorchroma_w[c_idx]
    c_chroma_d <- hz$colorchroma_d[c_idx]

    c_oc <- hz$oc[c_idx]

    cond5a_val_w <- cond4[cond4 %in% which(hz$colorvalue_w < c_value_w-1)]

    cond5a_chroma_w <- cond4[cond4 %in% which(hz$colorchroma_w <
c_chroma_w-2)]
```

```
    cond5a_chroma_d <- cond4[cond4 %in% which(hz$colorchroma_d <
c_chroma_d-2)]

    cond5a_chroma <- list(cond5a_chroma_w,cond5a_chroma_d)
    cond5a_chroma_hz1 <- unlist(lapply(cond5a_chroma, function(x) { 1 %in% x
}))
    cond5a_chroma <- cond5a_chroma[cond5a_chroma_hz1]
    cond5a_chroma <- cond5a_chroma[which.max(lapply(cond5a_chroma,length))]


    cond5a_idx <- list(cond5a_val_w,cond5a_chroma)
    cond5a_hz1 <- unlist(lapply(cond5a_idx, function(x) { 1 %in% x }))
    cond5a_idx <- cond5a_idx[cond5a_hz1]
    cond5a_idx <- cond5a_idx[which.max(lapply(cond5a_idx,length))]


    cond5a <- cond5a_idx[cond5a_idx %in% which(hz$oc > 1.006*c_oc)]

    if(length(cond5a)==0) {
      diags[["umbric"]]["cond5a"][diags[["umbric"]]["pedon"]==id,] <- FALSE
    } else {
      diags[["umbric"]]["cond5a"][diags[["umbric"]]["pedon"]==id,] <- TRUE
    }


  } else {
    cond5a <- vector()
    diags[["umbric"]]["cond5a"][diags[["umbric"]]["pedon"]==id,] <- FALSE
  }



  ## Cond 5c

  if(diags[["umbric"]]["cond5a"][diags[["umbric"]]["pedon"]==id,] == FALSE) {

    cond5b <- cond4[cond4 %in% which(hz$oc >= 0.6)]

    if(length(cond5b)==0) {
      diags[["umbric"]]["cond5b"][diags[["umbric"]]["pedon"]==id,] <- FALSE
      diags[["umbric"]]["hz"][diags[["umbric"]]["pedon"]==id,] <- FALSE
    } else {
      diags[["umbric"]]["cond5b"][diags[["umbric"]]["pedon"]==id,] <- TRUE
    }

  } else {
    diags[["umbric"]]["cond5b"][diags[["umbric"]]["pedon"]==id,] <- NA
  }


  cond5 <- list(cond5a,cond5b)
  cond5_hz1 <- unlist(lapply(cond5, function(x) { 1 %in% x }))
  cond5 <- cond5[cond5_hz1]
  cond5 <- cond5[which.max(lapply(cond5,length))]
```

```
  cond5 <- unlist(cond5)


  ## Cond 6: Minimum thickness
  ## Assumes all hz meeting criteria up to here are within epipedon

  ## Cond 6a: 25cm if 6a1, 6a2, OR 6a3



  ## Cond 6a

  if(length(cond5)!=0 & sum(hz$hzthk[cond5] >= 25)) {


    ## Cond 6a1
    cond6a1 <- cond5[cond5 %in% which(tolower(hz$texture) %in% c("loamy fine
sand","loamy sand","loamy coarse sand",
                                                        "very fine sand","fine
sand","sand","coarse sand",
                                                        "lfs","ls","lcos","vfs","fs","s","c

    if(sum(hz$hzthk[cond6a1]>=25)) {

      diags[["umbric"]]["cond6a1"][diags[["umbric"]]["pedon"]==id,] <- TRUE
      diags[["umbric"]]["hz"][diags[["umbric"]]["pedon"]==id,] <- TRUE
      diags[["umbric"]]["top_depth"][diags[["umbric"]]["pedon"]==id,] <-
hz$hzdept[1]
      diags[["umbric"]]["bottom_depth"][diags[["umbric"]]["pedon"]==id,] <-
max(hz$hzdepb[cond6a1])
      diags[["umbric"]]["thick"][diags[["umbric"]]["pedon"]==id,] <-
max(hz$hzdepb[cond6a1])

      return(diags)

    } else {

      diags[["umbric"]]["cond6a1"][diags[["umbric"]]["pedon"]==id,] <- FALSE

      ## cond6a2
      diag_sub <- c("agric", "albic", "anhydric", "argillic", "calcic",
                          "duripan", "fragipan", "glossic", "gypsic",
                          "kandic", "natric", "ortstein", "oxic",
"petrocalcic",
                          "petrogypsic", "placic", "salic", "sombric",
"spodic")

      if(!any(as.logical(lapply(diags[diag_sub],function(x){x[["hz"]]
[which(x[["pedon"]]==id)]})))) {

        ## Not checked: carbon content decreases irregularly with depth
        diags[["umbric"]]["cond6a2"][diags[["umbric"]]["pedon"]==id,] <- TRUE
        diags[["umbric"]]["hz"][diags[["umbric"]]["pedon"]==id,] <- TRUE
```

```
        diags[["umbric"]]["top_depth"][diags[["umbric"]]["pedon"]==id,] <-
hz$hzdept[1]
        diags[["umbric"]]["bottom_depth"][diags[["umbric"]]["pedon"]==id,] <-
max(hz$hzdepb[cond5])
        diags[["umbric"]]["thick"][diags[["umbric"]]["pedon"]==id,] <-
max(hz$hzdepb[cond5])

        return(diags)

      } else {



        diags[["umbric"]]["cond6a2"][diags[["umbric"]]["pedon"]==id,] <- FALSE

        ## cond 6a3

        ## Cond6a3a

        ## Not checked: secondary carbonates
        cond6a3a_hz <- c("calcic","petrocalcic","duripan","fragipan")
        if(any(as.logical(lapply(diags[cond6a3a_hz],function(x){x[["hz"]]
[which(x[["pedon"]]==id)]})))) {

          cond6a3a_hz_idx <-
which(as.logical(lapply(diags[cond6a3a_hz],function(x){x[["hz"]]
[which(x[["pedon"]]==id)]})))
          cond6a3a <- any(diags[[cond6a3a_hz[cond6a3a_hz_idx]]]
["bottom_depth"][diags[[cond6a3a_hz[cond6a3a_hz_idx]]]["pedon"]==id,] <=
75+diags[["orgmin"]]["min_top"][diags[["orgmin"]]["pedon"]==id,])

          if(cond6a3a) {

            diags[["umbric"]]["cond6a3a"][diags[["umbric"]]["pedon"]==id,] <-
TRUE

          } else {

            diags[["umbric"]]["cond6a3a"][diags[["umbric"]]["pedon"]==id,] <-
FALSE

          }


        } else {

          cond6a3a <- TRUE
          diags[["umbric"]]["cond6a3a"][diags[["umbric"]]["pedon"]==id,] <-
TRUE

        }


        ## Cond6a3b
```

```
        cond6a3b_hz <- c("argillic","natric","oxic","spodic")
        if(any(as.logical(lapply(diags[cond6a3b_hz],function(x){x[["hz"]]
[which(x[["pedon"]]==id)]})))) {

          cond6a3b_hz_idx <-
which(as.logical(lapply(diags[cond6a3b_hz],function(x){x[["hz"]]
[which(x[["pedon"]]==id)]})))
          cond6a3b <- max(diags[[cond6a3b_hz[cond6a3b_hz_idx]]]
["bottom_depth"][diags[[cond6a3b_hz[cond6a3b_hz_idx]]]["pedon"]==id,]) <=
75+diags[["orgmin"]]["min_top"][diags[["orgmin"]]["pedon"]==id,]


          if(cond6a3b) {

            diags[["umbric"]]["cond6a3b"][diags[["umbric"]]["pedon"]==id,] <-
TRUE

          } else {

            diags[["umbric"]]["cond6a3b"][diags[["umbric"]]["pedon"]==id,] <-
FALSE

          }

        } else {

          cond6a3b <- TRUE
          diags[["umbric"]]["cond6a3a"][diags[["umbric"]]["pedon"]==id,] <-
TRUE

        }


        ## final check for Cond6a3

        if(cond6a3a | cond6a3b) {

          diags[["umbric"]]["cond6a3"][diags[["umbric"]]["pedon"]==id,] <-
TRUE
          diags[["umbric"]]["hz"][diags[["umbric"]]["pedon"]==id,] <- TRUE
          diags[["umbric"]]["top_depth"][diags[["umbric"]]["pedon"]==id,] <-
hz$hzdept[1]
          diags[["umbric"]]["bottom_depth"][diags[["umbric"]]["pedon"]==id,]
<- max(hz$hzdepb[cond5])
          diags[["umbric"]]["thick"][diags[["umbric"]]["pedon"]==id,] <-
max(hz$hzdepb[cond5])

          return(diags)

        } else {

          diags[["umbric"]]["cond6a3"][diags[["umbric"]]["pedon"]==id,] <-
FALSE
```

```
        }


      }

    }




  }




  ## Cond 6b: 10cm if 6b

  ## Cond 6b

  if(length(cond5)!=0 & sum(hz$hzthk[cond5] >= 10)) {

    cond6b <- cond5[cond5 %in% which(!tolower(hz$texture) %in% c("loamy fine
sand","loamy sand","loamy coarse sand",
                                                          "very fine sand","fine
sand","sand","coarse sand",
                                                          "lfs","ls","lcos","vfs","fs","s","c

    if(length(cond6b)!=0) {

      cond6b_bottom <- max(hz$hzdepb[cond6b])
      cond6b_hz <- c("densic","lithic","paralithic","petrocalcic","duripan")
      cond6b_hz_idx <- which(as.logical(lapply(diags[cond6b_hz],function(x)
{x[["hz"]][which(x[["pedon"]]==id)]})))

      if(length(cond6b_hz_idx)!=0) {

        cond6b_hz_tops<- diags[[cond6b_hz[cond6b_hz_idx]]]["top_depth"]
[diags[[cond6b_hz[cond6b_hz_idx]]]["pedon"]==id,]

        if(any(cond6b_hz_tops == cond6b_bottom)) {

          diags[["umbric"]]["cond6b"][diags[["umbric"]]["pedon"]==id,] <- TRUE
          diags[["umbric"]]["hz"][diags[["umbric"]]["pedon"]==id,] <- TRUE
          diags[["umbric"]]["top_depth"][diags[["umbric"]]["pedon"]==id,] <-
hz$hzdept[1]
          diags[["umbric"]]["bottom_depth"][diags[["umbric"]]["pedon"]==id,]
<- cond6b_bottom
          diags[["umbric"]]["thick"][diags[["umbric"]]["pedon"]==id,] <-
cond6b_bottom
```

```
          return(diags)

        } else {

          diags[["umbric"]]["cond6b"][diags[["umbric"]]["pedon"]==id,] <-
FALSE

        }

      } else {

        diags[["umbric"]]["cond6b"][diags[["umbric"]]["pedon"]==id,] <- FALSE

      }


    } else {

      diags[["umbric"]]["cond6b"][diags[["umbric"]]["pedon"]==id,] <- FALSE

    }



  }




  ## Cond 6c: 18-25cm if 6c1 and/or 6c2


  cond6c1_hz <- c("calcic","petrocalcic","duripan","fragipan")
  cond6c2_hz <- c("argillic","natric","oxic","spodic")

  min_top <- dplyr::filter(diags[["orgmin"]],pedon==id)[["min_top"]]

  ## Cond 6c1
  cond6c1_hz_idx <- which(as.logical(lapply(diags[cond6c1_hz],function(x)
{x[["hz"]][which(x[["pedon"]]==id)]})))

  if(length(cond6c1_hz_idx)!=0 & sum(hz$hzthk[cond5]) >= 18) {

    cond6c1_thick <- min_top - min(diags[[cond6c1_hz[cond6c1_hz_idx]]]
["top_depth"][diags[[cond6c1_hz[cond6c1_hz_idx]]]["pedon"]==id,])


    if(sum(hz$hzthk[cond5]) >= 0.33*cond6c1_thick) {

      diags[["umbric"]]["cond6c1"][diags[["umbric"]]["pedon"]==id,] <- TRUE
      diags[["umbric"]]["hz"][diags[["umbric"]]["pedon"]==id,] <- TRUE
      diags[["umbric"]]["top_depth"][diags[["umbric"]]["pedon"]==id,] <-
hz$hzdept[1]
```

```
      diags[["umbric"]]["bottom_depth"][diags[["umbric"]]["pedon"]==id,] <-
sum(hz$hzthk[cond5])
      diags[["umbric"]]["thick"][diags[["umbric"]]["pedon"]==id,] <-
sum(hz$hzthk[cond5])

      return(diags)

    } else {

      diags[["umbric"]]["cond6c1"][diags[["umbric"]]["pedon"]==id,] <- FALSE

    }

  } else {

    diags[["umbric"]]["cond6c1"][diags[["umbric"]]["pedon"]==id,] <- FALSE

  }


  cond6c2_hz_idx <- which(as.logical(lapply(diags[cond6c2_hz],function(x)
{x[["hz"]][which(x[["pedon"]]==id)]})))
  if(length(cond6c2_hz_idx)!=0 & sum(hz$hzthk[cond5]) >= 18) {

    cond6c2_thick <- min_top - min(diags[[cond6c2_hz[cond6c2_hz_idx]]]
["top_depth"][diags[[cond6c2_hz[cond6c2_hz_idx]]]["pedon"]==id,])

    if(sum(hz$hzthk[cond5]) >= 0.33*cond6c2_thick) {

      diags[["umbric"]]["cond6c2"][diags[["umbric"]]["pedon"]==id,] <- TRUE
      diags[["umbric"]]["hz"][diags[["umbric"]]["pedon"]==id,] <- TRUE
      diags[["umbric"]]["top_depth"][diags[["umbric"]]["pedon"]==id,] <-
hz$hzdept[1]
      diags[["umbric"]]["bottom_depth"][diags[["umbric"]]["pedon"]==id,] <-
sum(hz$hzthk[cond5])
      diags[["umbric"]]["thick"][diags[["umbric"]]["pedon"]==id,] <-
sum(hz$hzthk[cond5])

      return(diags)

    }

  } else {

    diags[["umbric"]]["cond6c2"][diags[["umbric"]]["pedon"]==id,] <- FALSE

  }




  ## Cond 6d: 18cm if 7 AND 8
```

```
  ## CONDITION 7 NOT CHECKED

  cond8 <- cond5[cond5 %in% which(hz$n < 0.7)]
  if(length(cond8)!=0 & sum(hz$hzthk[cond8]) >= 18) {

    diags[["umbric"]]["cond8"][diags[["umbric"]]["pedon"]==id,] <- TRUE
    diags[["umbric"]]["hz"][diags[["umbric"]]["pedon"]==id,] <- TRUE
    diags[["umbric"]]["top_depth"][diags[["umbric"]]["pedon"]==id,] <-
hz$hzdept[1]
    diags[["umbric"]]["bottom_depth"][diags[["umbric"]]["pedon"]==id,] <-
sum(hz$hzthk[cond8])
    diags[["umbric"]]["thick"][diags[["umbric"]]["pedon"]==id,] <-
sum(hz$hzthk[cond8])

    return(diags)

  } else {

    diags[["umbric"]]["cond8"][diags[["umbric"]]["pedon"]==id,] <- FALSE
    diags[["umbric"]]["hz"][diags[["umbric"]]["pedon"]==id,] <- FALSE

  }

  ## CONDITION 9 NOT CHECKED UNTIL CODE FOR PLAGGEN IS WRITTEN

  return(diags)

}
```

```
################################################################################
# TODO:
# - Confirm top and bottom depth conditions
epi.ochric <- function(hzs, diags, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)

  epis <- c("anthropic","folistic","histic","melanic","mollic","umbric")

  if(!any(as.logical(lapply(diags[epis],function(x){x[["hz"]]
[which(x[["pedon"]]==id)]})))) {

    diags[["ochric"]]["hz"][diags[["ochric"]]["pedon"]==id,] <- TRUE


    illuv_hz <- c("argillic","kandic","natric","spodic")
```

```
    illuv_hz_idx <- which(as.logical(lapply(diags[illuv_hz],function(x)
{x[["hz"]][which(x[["pedon"]]==id)]})))

    if(length(illuv_hz_idx)!=0) {

      dhz <- illuv_hz[illuv_hz_idx]
      illuv_hz_top<- min(diags[[dhz]]["top_depth"][diags[[dhz]]
["pedon"]==id,])

      diags[["ochric"]]["top_depth"][diags[["ochric"]]["pedon"]==id,] <-
hz$hzdept[1]
      diags[["ochric"]]["bottom_depth"][diags[["ochric"]]["pedon"]==id,] <-
illuv_hz_top
      diags[["ochric"]]["thick"][diags[["ochric"]]["pedon"]==id,] <-
illuv_hz_top

    } else {

      if(any(grepl(hz$hzname,pattern="[Tp]"))) {

        pld <- max(hz$hzdepb[which(grepl(hz$hzname,pattern="[Tp]"))]) # plow
hz

      } else {

        pld <- 18

      }

      diags[["ochric"]]["top_depth"][diags[["ochric"]]["pedon"]==id,] <-
hz$hzdept[1]
      diags[["ochric"]]["bottom_depth"][diags[["ochric"]]["pedon"]==id,] <-
pld
      diags[["ochric"]]["thick"][diags[["ochric"]]["pedon"]==id,] <- pld

    }

    return(diags)

  } else {

    diags[["ochric"]]["hz"][diags[["ochric"]]["pedon"]==id,] <- FALSE

    return(diags)

  }


}


epi.epi <- function(diags, id) {
  # This function assigns the epipedon names.
  # Includes sanity check that there are not multiple epipedons.
```

```
  epis <-
c("anthropic","folistic","histic","melanic","mollic","umbric","ochric")

  epi <- epis[as.logical(lapply(diags[epis],function(x){x[["hz"]]
[which(x[["pedon"]]==id)]}))]



  ### Sanity check ###
  if(length(epi)>1) {

    stop(paste("Error: multiple epipedons calculated: ",paste(epi,collapse=',
'),sep=''))

  }

  if(length(epi)==0) {

    stop("Error: no epipedon calculated, check ochric conditions.")

  }


  top_depth <- diags[[epi]]["top_depth"][diags[[epi]]["pedon"]==id,]
  bottom_depth <- diags[[epi]]["bottom_depth"][diags[[epi]]["pedon"]==id,]
  thick <- bottom_depth - top_depth

  diags[["epipedon"]]["hz"][diags[["epipedon"]]["pedon"]==id,] <- epi
  diags[["epipedon"]]["top_depth"][diags[["epipedon"]]["pedon"]==id,] <-
top_depth
  diags[["epipedon"]]["bottom_depth"][diags[["epipedon"]]["pedon"]==id,] <-
bottom_depth
  diags[["epipedon"]]["thick"][diags[["epipedon"]]["pedon"]==id,] <- thick

  return(diags)


}
```

Appendix H:

subsurface_diag.R

```
# This script contains functions for walking through the logic
# for all subsurface diagnostic horizons from the Keys
#
# Main inputs: hz and diags
#
# depends on dplyr

# TODO:
# - ADD THICKNESS CALCULATIONS IN EACH FUNCTION
# - check that indices remain sequential
#   - which sequence to take?
# - check that top and bottom depths for each diags hz are recorded
# - record and reconcile all hz property columns
# - check secondary carbonates?
# - standardize code style and format
# - explicitly call dplyr dplyr::filter
# - what do we do with buried organic portions?
# - add glacic layer




##############################################################################
agric <- function(hzs, diags, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)

  cond1 <- which(hz$wormhole_vol > 5 & (hz$colorvalue_w <= 4 &
hz$colorchroma_w <= 2))

  cond2 <- which(hz$lamellae_vol >= 2.5 & hz$lamellae_thk >= 5 &
                 (hz$colorvalue_w <= 4 & hz$colorchroma_w <= 2))




  # checking cond1 with cond0

  if(length(cond1)!=0 & length(cond1[which(cond1 != 1)]) !=0 ) {


    cond1 <- cond1[which(cond1 != 1)]

    if((hz$hzname[min(cond1)-1] %in% c("A","Ap")) &
       sum(hz$hzthk[cond1]) >= 10) {

      diags[["agric"]]["cond1"][diags[["agric"]]["pedon"]==id,] <- TRUE

    } else {

      diags[["agric"]]["cond0"][diags[["agric"]]["pedon"]==id,] <- FALSE

    }
```

```
  } else {

    diags[["agric"]]["cond1"][diags[["agric"]]["pedon"]==id,] <- FALSE

  }




  # checking cond2 with cond0

  if(length(cond2)!=0 & length(cond1[which(cond2 != 1)]) !=0 ) {


    cond2 <- cond2[which(cond2 != 1)]

    if((hz$hzname[min(cond2)-1] %in% c("A","Ap")) &
       sum(hz$hzthk[cond1]) >= 10) {

      diags[["agric"]]["cond2"][diags[["agric"]]["pedon"]==id,] <- TRUE

    } else {

      diags[["agric"]]["cond0"][diags[["agric"]]["pedon"]==id,] <- FALSE

    }


  } else {

    diags[["agric"]]["cond2"][diags[["agric"]]["pedon"]==id,] <- FALSE

  }



  if(length(cond1)!=0 | length(cond2)!=0) {

    thk <- c(cond1=sum(hz$hzthk[cond1]), cond2=sum(hz$hzthk[cond2]))
    cond <- get(names(which.max(thk)))

    diags[["agric"]]["hz"][diags[["agric"]]["pedon"]==id,] <- TRUE
    diags[["agric"]]["top_depth"][diags[["agric"]]["pedon"]==id,] <-
min(hz$hzdept[cond])
    diags[["agric"]]["bottom_depth"][diags[["agric"]]["pedon"]==id,] <-
max(hz$hzdepb[cond])
    diags[["agric"]]["thick"][diags[["agric"]]["pedon"]==id,] <-
max(hz$hzdepb[cond]) - min(hz$hzdept[cond])

  } else {

    diags[["agric"]]["hz"][diags[["agric"]]["pedon"]==id,] <- FALSE
```

```
  }


  return(diags)

}









################################################################################
# TODO:
# - better check for consecutive hz that are actually albic
albic <- function(hzs, diags, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)

  albic_mat <- diags[["albic_mat"]]["prop"][diags[["albic_mat"]]
["pedon"]==id,]

  if(albic_mat) {

    albic_idx <- hz$albic_vol>85 & grepl(hz$desgnmaster,pattern="[E]")

    if(any(albic_idx)) {
      diags[["albic"]]["hz"][diags[["albic"]]["pedon"]==id,] <- TRUE
      diags[["albic"]]["top_depth"][diags[["albic"]]["pedon"]==id,] <-
min(hz$hzdept[albic_idx])
      diags[["albic"]]["bottom_depth"][diags[["albic"]]["pedon"]==id,] <-
max(hz$hzdepb[albic_idx])
      diags[["albic"]]["thick"][diags[["albic"]]["pedon"]==id,] <-
max(hz$hzdepb[albic_idx]) - min(hz$hzdept[albic_idx])
    } else {
      diags[["albic"]]["hz"][diags[["albic"]]["pedon"]==id,] <- FALSE
    }



  } else {

    diags[["albic"]]["hz"][diags[["albic"]]["pedon"]==id,] <- FALSE

  }

  return(diags)
```

```
}
```

```
################################################################################
# TODO:
# - check condition 5: calcium sulfate minerals
anhydric <- function(hzs, diags, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)

  cond2 <- which(hz$anhyd >= 5)

  if(length(cond2)==0) {

    diags[["anhydric"]]["cond2"][diags[["anhydric"]]["pedon"]==id,] <- FALSE
    diags[["anhydric"]]["hz"][diags[["anhydric"]]["pedon"]==id,] <- FALSE

    return(diags)

  } else {

    diags[["anhydric"]]["cond2"][diags[["anhydric"]]["pedon"]==id,] <- TRUE

  }

  cond3 <- cond2[which(hz$colorhue_d == "5Y" &
                       hz$colorhue_w == "5Y" &
                       hz$colorchroma_d %in% c(1,2) &
                       hz$colorchroma_w %in% c(1,2) &
                       hz$colorvalue_d %in% c(7,8) &
                       hz$colorvalue_w %in% c(7,8)) %in% cond2]


  if(length(cond3)==0) {

    diags[["anhydric"]]["cond3"][diags[["anhydric"]]["pedon"]==id,] <- FALSE
    diags[["anhydric"]]["hz"][diags[["anhydric"]]["pedon"]==id,] <- FALSE

    return(diags)

  } else {

    diags[["anhydric"]]["cond3"][diags[["anhydric"]]["pedon"]==id,] <- TRUE

  }
```

```
  if(sum(hz$hzthk[cond3]) >= 15) {

    wa_anhyd <- weighted.mean(hz$anhyd[cond3],hz$hzthk[cond3]/
sum(hz$hzthk[cond3]))

    if(wa_anhyd * sum(hz$hzthk[cond3]) >= 150) {

      diags[["anhydric"]]["cond1"][diags[["anhydric"]]["pedon"]==id,] <- TRUE
      diags[["anhydric"]]["cond4"][diags[["anhydric"]]["pedon"]==id,] <- TRUE

      ## condition 5 NOT CHECKED

      diags[["anhydric"]]["hz"][diags[["anhydric"]]["pedon"]==id,] <- TRUE
      diags[["anhydric"]]["top_depth"][diags[["anhydric"]]["pedon"]==id,] <-
min(hz$hzdept[cond3])
      diags[["anhydric"]]["bottom_depth"][diags[["anhydric"]]["pedon"]==id,]
<- max(hz$hzdept[cond3])
      diags[["anhydric"]]["thick"][diags[["anhydric"]]["pedon"]==id,] <-
max(hz$hzdept[cond3]) - min(hz$hzdept[cond3])


    } else {

      diags[["anhydric"]]["cond4"][diags[["anhydric"]]["pedon"]==id,] <- FALSE
      diags[["anhydric"]]["hz"][diags[["anhydric"]]["pedon"]==id,] <- FALSE

    }


  } else {

    diags[["anhydric"]]["cond1"][diags[["anhydric"]]["pedon"]==id,] <- FALSE
    diags[["anhydric"]]["hz"][diags[["anhydric"]]["pedon"]==id,] <- FALSE

  }

  return(diags)

}
```

```
##############################################################################
```

```
# TODO:
# - edit code to better record conditions
argillic <- function(hzs, diags, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)

  # Gets indices of horizons with "t" subordinate designation
  # Used to generally satisfy all components of cond1b
  cond1b <- grep(hz$hzname,pattern="[Tt]")

  if(length(cond1b)!=0) {
    diags[["argillic"]]["cond1b"][diags[["argillic"]]["pedon"]==id,] <- TRUE
  }


  ## The following is adapted from AQP ##

  hzdesgn <- 'hzname'
  require_t <- TRUE
  bottom.pattern <- "Cr|R|Cd"
  lower.grad.pattern <- "^[2-9]*B*CB*[^rtd]*[1-9]*$"
  #sandy.texture.pattern <- "-S$|^S$|COS$|L[^V]FS$|[^L]VFS$|LS$|LFS$"
  sandy.texture.pattern <- c("sand","coarse sand","loamy coarse sand",
                             "loamy fine sand","loamy sand")
  verbose <- FALSE

  depthcol <- c("hzdept","hzdepb")


  has_t <- grepl(as.character(hz$hzname), pattern = "[Tt]")
  pld <- hz$hzdepb[grep(hz$hzname,pattern="[Tp]")]
  mss <- dplyr::filter(diags[["orgmin"]],pedon==id)[["min_top"]]

  if(length(pld)==0) {
    pld <- FALSE
  } else if(length(pld)>1) {
    pld <- pld[length(pld)] # if more than 1 Ap
  }

  # elluv <- shallowest_t-1 # index of the elluvial hz above the first t
  # elluv_clay <- hz$claytotal[elluv]

  upper.bound <- argillic.clay.increase.depth(hz,"claytotal")
  lower.bound <- -Inf
  soil.depth.idx <- (hz$densic | hz$lithic | hz$paralithic)
  if(any(soil.depth.idx)) {
    soil.depth <- min(hz$hzdept[soil.depth.idx])
  } else {
    soil.depth <- max(hz$hzdepb)
  }



  if(is.na(upper.bound)) {
```

```
    #return(c(ubound = NA, lbound = NA))
    diags[["argillic"]]["hz"][diags[["argillic"]]["pedon"]==id,] <- FALSE
    return(diags)
  }


  ## ensure upper bound is in t hz
  if(length(upper.bound)!=0 & !is.na(upper.bound)) {

    if(!grepl(hz$hzname[which(hz$hzdept == upper.bound)],pattern="[Tt]")) {

      if(length(cond1b)!=0) {
        shallowest_t <- min(hz$hzdept[cond1b]) # index of the first t
        upper.bound <- shallowest_t
      } else {
        shallowest_t <- NA
      }
    }
  }




  ## this doesn't make sense; if upper.bound is not given, there is no
argillic
  if(is.na(upper.bound)) {

    if(length(cond1b)!=0) {
      shallowest_t <- min(hz$hzdept[cond1b]) # index of the first t
    } else {
      shallowest_t <- NA
    }


    depth_ld <- hz$hzdept[grepl(hz$hzname, pattern = "^[2-9].*[Tt]")]

    if (!is.na(shallowest_t) & shallowest_t != Inf) {
      if (any(pld == shallowest_t)) {
        upper.bound <- shallowest_t
      }

      if (!all(is.na(depth_ld))) {
        if (any(shallowest_t %in% depth_ld)) {
          upper.bound <- shallowest_t
        }
      }
    }
  }



  ## Section adapted from AQP ##

  if (!is.na(upper.bound)) {
```

```
    ########
    # TODO: allow evidence of illuviation from lab data fine clay ratios etc?
how?
    #
    # TODO: if `require_t` is set to `FALSE`... or in a future
getKandicBounds()...
    #       how do you detect the bottom of a argillic or kandic horizon
(which need not have clay films)
    #          in e.g. saprolite ?
    #       could you look for C in master horizon designation for e.g. rock
structure / parent material?
    #       in lieu of checking for t and a cemented bedrock contact... is
that how lower.bound should be determined?
    ########
    if (sum(has_t) | !require_t) {
      # get index of last horizon with t
      idx.last <- rev(which(has_t))[1]

      ## Partial fix for TODO #2? seems reasnable for the require_t=FALSE
lower.bound case
      # take _very_ last horizon depth first (will be truncated to contact
depth if needed)
      depth.last <- as.numeric(aqp:::.data.frame.j(hz[nrow(hz),], depthcol[2],
"data.frame"))

      # take the top depth of any B or C horizon  without t subscript above
"depth.last"
      c.idx <- which(grepl(hz[[hzdesgn]], pattern = lower.grad.pattern))
      if (length(c.idx)) {
        c.horizon <- as.numeric(aqp:::.data.frame.j(hz[c.idx[1], ],
depthcol[1], "data.frame"))

        # if the _shallowest C horizon_ top depth is above the _last horizon_
bottom depth (could be top depth of same hz)
        if (c.horizon < depth.last)  {
          # use the top depth of the first C horizon that matched the pattern
          if (verbose) {
            message(paste0("Found ",paste0(hz[[hzdesgn]][c.idx], collapse =
","),
                           " below argillic, adjusting lower bound (",
                           "pedon_id",": ", as.character(id),")"))
            # plot(p)
            # print(c.idx)
            depth.last <- c.horizon
          }
        }
      }

      # get the bottom depth of the last horizon with a t (this could be same
as c.horizon above)
      if (require_t) {
        depth.last <- as.numeric(aqp:::.data.frame.j(hz[idx.last,],
                                                     depthcol[2],
```

```
                                                "data.frame"))
      }

      # in rare cases, the bottom depth of the bedrock/contact is not
populated
      # step back until we find one that is not NA
      idx.last.i <- idx.last
      while (is.na(depth.last) & idx.last.i >= 0) {
        depth.last <- as.numeric(aqp:::.data.frame.j(hz[idx.last.i,],
                                                depthcol[2],
                                                "data.frame"))
        idx.last.i <- idx.last.i - 1
      }

      # if the last horizon with a t is below the contact (Crt or Rt) or some
other weird reason
      if (soil.depth < depth.last) {
        # return the soil depth to contact
        lower.bound <- soil.depth

      } else {
        #otherwise, return the bottom depth of the last horizon with a t
        lower.bound <- depth.last
      }
    } else {
      if (verbose) {
        message(paste0("Profile (",as.character(id),") has clay increase with
no evidence of illuviation (t)."))
      }
      lower.bound <- NA
      upper.bound <- NA
    }
  } else {

    # if the upper bound is NA, return NA for the lower bound
    lower.bound <- NA
  }

  if (!is.finite(lower.bound)) {
    lower.bound <- NA
  }


  if(!is.na(upper.bound)) {
    #bdepthspc <- glom(p, mss, upper.bound, df = TRUE)
    bdepthspc <- glom_br(hz=hz, pedon_id=id, top=mss, bottom=upper.bound)
  } else {
    bdepthspc <- NA
  }


  # if there are no overlying horizons, return NA
  if (is.null(bdepthspc) | all(is.na(bdepthspc))) {
    if (verbose) {
```

```
      message(paste0("Profile (",id,
                     ") has no horizons overlying a [possible] argillic."))
      #return(c(ubound = NA, lbound = NA))

    }
    diags[["argillic"]]["hz"][diags[["argillic"]]["pedon"]==id,] <- FALSE
  }


  if (all(!is.na(c(upper.bound, lower.bound)))) {
    bdepths <- bdepthspc[[depthcol[2]]]

    # if argi bounds are found check that minimum thickness requirements are
met
    min.thickness <- max(7.5, max(bdepths, na.rm = TRUE) / 10)

    #textures <- glom(p, upper.bound, lower.bound, df = TRUE)[[texcl.attr]]
    textures <- glom_br(hz=hz, top=upper.bound, bottom=lower.bound)
[["texture"]]
    # is_sandy <- all(grepl(sandy.texture.pattern, textures, ignore.case =
TRUE))
    is_sandy <- all(textures %in% sandy.texture.pattern)

    if (is_sandy) {
      min.thickness <- 15
    }

    if (lower.bound - upper.bound < min.thickness) {
      if (verbose) {
        message(paste0("Profile (",id,
                       ") does not meet thickness requirement."))
        #return(c(ubound = NA, lbound = NA))
      }
      diags[["argillic"]]["hz"][diags[["argillic"]]["pedon"]==id,] <- FALSE
      diags[["argillic"]]["cond1a"][diags[["argillic"]]["pedon"]==id,] <-
FALSE
    }
  }

  if (!is.na(upper.bound)) {
    # it is possible that a subhorizon of the Ap horizon meets the clay
increase
    if (pld > upper.bound) {
      upper.bound <- pld
      if (verbose) {
        message(paste0("Profile (",id,
                       ") meets clay increase within plowed layer."))

      }

    }

    diags[["argillic"]]["cond2"][diags[["argillic"]]["pedon"]==id,] <- TRUE
    diags[["argillic"]]["hz"][diags[["argillic"]]["pedon"]==id,] <- TRUE
```

```
    diags[["argillic"]]["top_depth"][diags[["argillic"]]["pedon"]==id,] <-
upper.bound
    diags[["argillic"]]["bottom_depth"][diags[["argillic"]]["pedon"]==id,] <-
lower.bound
    diags[["argillic"]]["thick"][diags[["argillic"]]["pedon"]==id,] <-
lower.bound - upper.bound

  } else {
    diags[["argillic"]]["cond2"][diags[["argillic"]]["pedon"]==id,] <- FALSE
  }


  #return(c(ubound = upper.bound, lbound = lower.bound))
  return(diags)


}









################################################################################
# TODO:
# - need to determine partclass before running this?
# - wording implies the calcic horizon meets particle size criteria
# - need to add frags up to 75mm to hzs cols
# - consider changing to clay sized carbonates
# - confirm the meaning of cond2a
# - assumes CaCO3 at least 5% if 'k'
# - assumes CaCO3 at least 50% if 'kk
calcic <- function(hzs, diags, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)



  ## cond 1 AND 2 (2a OR 2b OR 2c (2c1 AND 2c2 AND 2c3)) AND 3 (3a OR 3b OR
3c)

  # cond 2a

  cond2a_15 <- which(hz$caco3 >= 15)

  if(length(cond2a_15)!=0) {

    wa_caco3 <- weighted.mean(hz$caco3[cond2a_15], hz$hzthk[cond2a_15] /
hz$hzthk[cond2a_15])
```

```
  if(length(cond2a_15)>1 & nrow(hz) %in% cond2a_15) {

    cond2a_und <- nrow(hz)



    if(wa_caco3 >= 5+hz$caco3[cond2a_und]) {

      cond2a <- cond2a_15
      diags[["calcic"]]["cond2a"][diags[["calcic"]]["pedon"]==id,] <- TRUE

    } else {

      cond2a <- vector()
      diags[["calcic"]]["cond2a"][diags[["calcic"]]["pedon"]==id,] <- FALSE

    }



  } else if(length(cond2a_15)>1) {

    cond2a_und <- max(cond2a_15)+1


    if(wa_caco3 >= 5+hz$caco3[cond2a_und]) {

      cond2a <- cond2a_15
      diags[["calcic"]]["cond2a"][diags[["calcic"]]["pedon"]==id,] <- TRUE

    } else {

      cond2a <- vector()
      diags[["calcic"]]["cond2a"][diags[["calcic"]]["pedon"]==id,] <- FALSE

    }



  } else {

    cond2a <- vector()

  }



} else {

  cond2a <- vector()
```

```
  }




  # Cond 2b
  cond2b_15 <- which(hz$caco3 >= 15)
  cond2b <- cond2b_15[cond2b_15 %in% which(hz$seccarb >= 5)]

  if(length(cond2b)==0) {
    cond2b <- vector()
  }




  # Cond2c

  cond2c <- which(hz$caco3 >= 5)

  if(length(cond2c)==0) {

    diags[["calcic"]]["cond2c"][diags[["calcic"]]["pedon"]==id,] <- FALSE
    diags[["calcic"]]["cond2c1"][diags[["calcic"]]["pedon"]==id,] <- FALSE
    diags[["calcic"]]["cond2c2"][diags[["calcic"]]["pedon"]==id,] <- FALSE
    diags[["calcic"]]["cond2c3"][diags[["calcic"]]["pedon"]==id,] <- FALSE

  } else {

    cond2c1 <- cond2c[cond2c %in% which(hz$claytotal <= 18)]
    cond2c2_part <- all(hz$texture[cond2c1] %in% c("coarse sand","sand","fine
sand",
                                                   "loamy coarse sand","loamy
sand",
                                                   "loamy fine sand")) |
                        all(hz$fragtotvol[cond2c1] >= 35 &
                              hz$texture[cond2c1] %in% c("coarse
sand","sand","fine sand",
                                                         "loamy coarse
sand","loamy sand",
                                                         "loamy fine sand")) |
                        all(hz$claytotal[cond2c1]<=18 &
                              (hz$sandtotal[cond2c1]+hz$fragtotvol[cond2c1]>=15))
|
                        all(hz$fragtotvol[cond2c1]>=35 &
hz$claytotal[cond2c1]<=35)

    if(length(cond2c1)==0) {

      diags[["calcic"]]["cond2c"][diags[["calcic"]]["pedon"]==id,] <- FALSE
      diags[["calcic"]]["cond2c1"][diags[["calcic"]]["pedon"]==id,] <- FALSE
```

```
    } else if(cond2c2_part) {

      cond2c2 <- cond2c1

      cond2c3_2carb <- cond2c2[cond2c2 %in% which(hz$seccarb >= 5)]

      if(nrow(hz) %in% cond2c2) {

        if(length(cond2c2)>1) {
          cond2c3_5caco3 <- hz$caco3[max(cond2c2)-1] >=
5+hz$caco3[max(cond2c2)]
        } else {
          cond2c3_5caco3 <- FALSE
        }

      } else {
        cond2c3_5caco3 <- hz$caco3[max(cond2c2)] >= 5+hz$caco3[max(cond2c2)+1]
      }



      if(length(cond2c3_2carb) > length(cond2c2)) {

        cond2c <- cond2c3_2carb

      } else if(cond2c3_5caco3) {

        cond2c <- cond2c2

      } else {

        cond2c <- vector()
        diags[["calcic"]]["cond2c"][diags[["calcic"]]["pedon"]==id,] <- FALSE
        diags[["calcic"]]["cond2c3"][diags[["calcic"]]["pedon"]==id,] <- FALSE

      }


    } else {

      cond2c <- vector()
      diags[["calcic"]]["cond2c"][diags[["calcic"]]["pedon"]==id,] <- FALSE
      diags[["calcic"]]["cond2c2"][diags[["calcic"]]["pedon"]==id,] <- FALSE

    }


  }



  # Pick which of 2a, 2b, and 2c is bigger
```

```
  cond2 <- list(cond2a,cond2b,cond2c)
  cond2 <- unlist(cond2[which.max(lapply(cond2,length))])

  if(length(cond2)==0) {

    diags[["calcic"]]["cond2"][diags[["calcic"]]["pedon"]==id,] <- FALSE
    diags[["calcic"]]["hz"][diags[["calcic"]]["pedon"]==id,] <- FALSE

    return(diags)

  }


  ## Only checking conditions 3 and 3c
  if(any(hz$cemented[cond2])) {

    cem <- cond2[cond2 %in% which(hz$cemented)]
    if(any(hz$hzthk[cem] > 10)) {

      diags[["calcic"]]["cond3"][diags[["calcic"]]["pedon"]==id,] <- FALSE
      diags[["calcic"]]["cond3c"][diags[["calcic"]]["pedon"]==id,] <- FALSE
      diags[["calcic"]]["hz"][diags[["calcic"]]["pedon"]==id,] <- FALSE

      return(diags)

    } else {

      cond3 <- TRUE
      cond3c <- TRUE
      diags[["calcic"]]["cond3"][diags[["calcic"]]["pedon"]==id,] <- TRUE
      diags[["calcic"]]["cond3c"][diags[["calcic"]]["pedon"]==id,] <- TRUE

    }

  } else {

    cond3 <- TRUE
    diags[["calcic"]]["cond3"][diags[["calcic"]]["pedon"]==id,] <- TRUE

  }


  # checking Condition 1: thickness requirement
  if(sum(hz$hzthk[cond2]) > 15 & cond3) {

    diags[["calcic"]]["cond1"][diags[["calcic"]]["pedon"]==id,] <- TRUE
    diags[["calcic"]]["hz"][diags[["calcic"]]["pedon"]==id,] <- TRUE
    diags[["calcic"]]["top_depth"][diags[["calcic"]]["pedon"]==id,] <-
min(hz$hzdept[cond2])
    diags[["calcic"]]["bottom_depth"][diags[["calcic"]]["pedon"]==id,] <-
max(hz$hzdepb[cond2])
    diags[["calcic"]]["thick"][diags[["calcic"]]["pedon"]==id,] <-
max(hz$hzdepb[cond2]) - min(hz$hzdept[cond2])
```

```
  } else {

    diags[["calcic"]]["cond1"][diags[["calcic"]]["pedon"]==id,] <- FALSE
    diags[["calcic"]]["hz"][diags[["calcic"]]["pedon"]==id,] <- FALSE

  }


  return(diags)



}









################################################################################
# TODO:
# - revert back to original implementation with better assumptions
# - removed epipedons from cond3 diags check
# - need to filter cond3 to start at bottom of epipedons and where no other
diags
# - consider changing aquic conditions to diags
# - add cond 2a2, cond 2b2
# - add check for redder hue in cond 2b1
cambic <- function(hzs, diags, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)


  #### New Cambic Implementation based on B hz and lack of diagnostics ####

  t <- diag_top_depth(c("duripan","fragipan","argillic","calcic","gypsic",
                        "natric","oxic","petrocalcic","petrogypsic","placic",
                        "salic","spodic","sulfuric"),diags,id)

  # b <- diag_bot_depth(c("duripan","fragipan","argillic","calcic","gypsic",
  #
"natric","oxic","petrocalcic","petrogypsic","placic",
  #                       "salic","spodic","sulfuric"),diags,id)

  ## ignore ochric bottom for thickness check
  if(diags[["epipedon"]]["hz"][diags[["epipedon"]]["pedon"]==id,] != "ochric")
{
```

```
    epi_top <- diags[["epipedon"]]["top_depth"][diags[["epipedon"]]
["pedon"]==id,]
    epi_bot <- diags[["epipedon"]]["bottom_depth"][diags[["epipedon"]]
["pedon"]==id,]

    # epi_sl <- prof_slice(epi_bot,epi_top,hz)

  } else {

    epi_top <- 0
    epi_bot <- 0

  }


  if(any(is.na(t))) {

    sl <- prof_slice(max(hz$hzdepb),epi_bot,hz)

    cond0b <- grep('[B]',sl$desgnmaster)

    cond1 <- cond0b[cond0b %in% which(tolower(sl$texture) %in% c("very fine
sand","loamy very fine sand","coarse sandy loam",
                                                "sandy loam","fine sandy
loam","very fine sandy loam","loam",
                                                "silt loam","silt","sandy clay
loam","clay loam","silty clay loam",
                                                "sandy clay","silty
clay","clay") &
                      sl$hzname != "Ap")]

    if(length(cond1)!=0) {

      diags[["cambic"]]["cond0b"][diags[["cambic"]]["pedon"]==id,] <- TRUE

      if(sum(sl$hzthk[cond1]) >= 15) {

        diags[["cambic"]]["cond0"][diags[["cambic"]]["pedon"]==id,] <- TRUE
        diags[["cambic"]]["hz"][diags[["cambic"]]["pedon"]==id,] <- TRUE
        diags[["cambic"]]["top_depth"][diags[["cambic"]]["pedon"]==id,] <-
min(sl$hzdept[cond1])
        diags[["cambic"]]["bottom_depth"][diags[["cambic"]]["pedon"]==id,] <-
max(sl$hzdepb[cond1])
        diags[["cambic"]]["thick"][diags[["cambic"]]["pedon"]==id,] <-
max(sl$hzdepb[cond1]) - min(sl$hzdept[cond1])
        return(diags)

      } else {

        diags[["cambic"]]["cond0b"][diags[["cambic"]]["pedon"]==id,] <- FALSE
        diags[["cambic"]]["hz"][diags[["cambic"]]["pedon"]==id,] <- FALSE

        return(diags)
```

```
    }


  } else {

    diags[["cambic"]]["cond0b"][diags[["cambic"]]["pedon"]==id,] <- FALSE
    diags[["cambic"]]["hz"][diags[["cambic"]]["pedon"]==id,] <- FALSE

    return(diags)

  }

}


#### Old Cambic Implementation based on Keys ####

# ## internal function to check cambic conditions
# cambic_conds <- function(sl,epi_sl,diags) {
#
#
#   # cond 1: must have texture
#   cond1 <- which(tolower(sl$texture) %in% c("very fine sand","loamy very
fine sand","coarse sandy loam",
#                                             "sandy loam","fine sandy
loam","very fine sandy loam","loam",
#                                             "silt loam","silt","sandy clay
loam","clay loam","silty clay loam",
#                                             "sandy clay","silty
clay","clay") &
#                    sl$hzname != "Ap")
#
#
#
#
#   if(length(cond1)==0) {
#     diags[["cambic"]]["cond1"][diags[["cambic"]]["pedon"]==id,] <- FALSE
#     diags[["cambic"]]["hz"][diags[["cambic"]]["pedon"]==id,] <- FALSE
#     return(diags)
#   }
#
#   diags[["cambic"]]["cond1"][diags[["cambic"]]["pedon"]==id,] <- TRUE
#
#
#
#
#
#
#
#   # cond 2: evidence of alteration; a (a1 AND a2 AND a3) OR b (b1 OR b2)
#
#   # aquic conditions plus 2a1, 2a2, 2a3
#   cond2a <- cond1[cond1 %in% which(sl$hzdepb <= 50 & sl$aquic)]
#
#   if(length(cond2a)!=0) {
```

```
  #
  #
  #     aquic_thk <- sum(sl$hzthk[cond2a])
  #     cond2a1_struct <- cond2a[cond2a %in% which(!sl$structtype %in%
c("massive","single grain"))]
  #     struct_thk <- sum(sl$hzthk[cond2a1_struct])
  #     cond2a1 <- (struct_thk / aquic_thk) >= 0.5
  #
  #     ## cond2a2 not checked ##
  #
  #     cond2a3a <- cond2a[cond2a %in% which(sl$colorvalue_w <= 3 &
  #                                          sl$colorchroma_w == 0 &
  #                                          sl$colorhue_w == "N")]
  #
  #     cond2a3b <- cond2a[cond2a %in% which(sl$colorvalue_w >= 4 &
sl$colorchroma_w <= 1)]
  #
  #     cond2a3c <- cond2a[cond2a %in% which(sl$colorchroma_w <= 2 &
sl$redoxconcen)]
  #
  #     cond2a3 <- list(cond2a3a,cond2a3b,cond2a3c)
  #     cond2a3 <- unlist(cond2a3[which.max(lapply(cond2a3,length))])
  #
  #     if(cond2a1 & length(cond2a3)!=0) {
  #
  #       diags[["cambic"]]["cond2a1"][diags[["cambic"]]["pedon"]==id,] <-
TRUE
  #       diags[["cambic"]]["cond2a3"][diags[["cambic"]]["pedon"]==id,] <-
TRUE
  #
  #       if(sum(sl$hzthk[cond2a3]) >= 15) {
  #
  #         diags[["cambic"]]["cond0"][diags[["cambic"]]["pedon"]==id,] <-
TRUE
  #         diags[["cambic"]]["hz"][diags[["cambic"]]["pedon"]==id,] <- TRUE
  #         diags[["cambic"]]["top_depth"][diags[["cambic"]]["pedon"]==id,] <-
min(sl$hzdept[cond2a3])
  #         diags[["cambic"]]["bottom_depth"][diags[["cambic"]]["pedon"]==id,]
<- max(sl$hzdepb[cond2a3])
  #         return(diags)
  #
  #
  #       }
  #
  #
  #
  #     }
  #
  #
  #   } else {
  #
  #     diags[["cambic"]]["cond2a"][diags[["cambic"]]["pedon"]==id,] <- FALSE
  #     diags[["cambic"]]["cond2a1"][diags[["cambic"]]["pedon"]==id,] <- FALSE
  #     diags[["cambic"]]["cond2a2"][diags[["cambic"]]["pedon"]==id,] <- FALSE
```

```
  #     diags[["cambic"]]["cond2a3"][diags[["cambic"]]["pedon"]==id,] <- FALSE
  #
  #
  #     cond1_thk <- sum(sl$hzthk[cond1])
  #     cond1_struct <- cond1[cond1 %in% which(!sl$structtype %in%
c("massive","single grain") | sl$fragtotvol == 0)]
  #     struct_thk <- sum(sl$hzthk[cond1_struct])
  #     cond2b_struct <- (struct_thk / cond1_thk) >= 0.5
  #
  #
  #     if(cond2b_struct) {
  #
  #       ## redder hue not checked ##
  #
  #       wa_chroma <- weighted.mean(sl$colorchroma_w[cond1],sl$hzthk[cond1]/
sum(sl$hzthk[cond1]))
  #       wa_value <- weighted.mean(sl$colorvalue_w[cond1],sl$hzthk[cond1]/
sum(sl$hzthk[cond1]))
  #       wa_clay <- weighted.mean(sl$claytotal[cond1],sl$hzthk[cond1]/
sum(sl$hzthk[cond1]))
  #
  #       # check not bottom of pedon
  #       if(max(sl$hzdepb[cond1])!=max(hz$hzdepb)) {
  #
  #         cond2b1_below <- (wa_chroma > sl$colorchroma_w[max(cond1)+1]) |
  #           (wa_value > sl$colorvalue_w[max(cond1)+1]) |
  #           (wa_clay > sl$claytotal[max(cond1+1)])
  #
  #       } else {
  #
  #         cond2b1_below <- FALSE
  #
  #       }
  #
  #       # check overlaying hz in epi
  #       if(min(sl$hzdept[cond1])!=min(epi_sl$hzdept)) {
  #
  #         cond2b1_above <- (wa_chroma > epi_sl$colorchroma_w[nrow(epi_sl)])
|
  #           (wa_value > epi_sl$colorvalue_w[nrow(epi_sl)]) |
  #           (wa_clay > epi_sl$claytotal[nrow(epi_sl)])
  #
  #       } else {
  #
  #         cond2b1_above <- FALSE
  #
  #       }
  #
  #
  #       ## cond2b2 not checked ##
  #
  #
  #       if(any(cond2b1_above,cond2b1_below)) {
  #
```

```
  #         diags[["cambic"]]["cond2b1"][diags[["cambic"]]["pedon"]==id,] <-
TRUE
  #
  #         if(sum(sl$hzthk[cond1]) >= 15) {
  #
  #           diags[["cambic"]]["cond0"][diags[["cambic"]]["pedon"]==id,] <-
TRUE
  #           diags[["cambic"]]["hz"][diags[["cambic"]]["pedon"]==id,] <- TRUE
  #           diags[["cambic"]]["top_depth"][diags[["cambic"]]["pedon"]==id,]
<- min(sl$hzdept[cond1])
  #           diags[["cambic"]]["bottom_depth"][diags[["cambic"]]
["pedon"]==id,] <- max(sl$hzdepb[cond1])
  #           diags[["cambic"]]["thick"][diags[["cambic"]]["pedon"]==id,] <-
max(sl$hzdepb[cond1]) - min(sl$hzdept[cond1])
  #           return(diags)
  #
  #
  #         }
  #
  #
  #       }
  #
  #
  #       # else not needed; return within if
  #       diags[["cambic"]]["cond2b"][diags[["cambic"]]["pedon"]==id,] <-
FALSE
  #       diags[["cambic"]]["hz"][diags[["cambic"]]["pedon"]==id,] <- FALSE
  #       return(diags)
  #
  #
  #
  #
  #
  #
  #     } else {
  #
  #       diags[["cambic"]]["cond2b"][diags[["cambic"]]["pedon"]==id,] <-
FALSE
  #       diags[["cambic"]]["hz"][diags[["cambic"]]["pedon"]==id,] <- FALSE
  #       return(diags)
  #
  #     }
  #
  #
  #
  #
  #   }
  #
  #   return(diags)
  #
  #
  # }
  #
  #
```

```
  # ## cond3
  #
  # t <- diag_top_depth(c("duripan","fragipan","argillic","calcic","gypsic",
  #
"natric","oxic","petrocalcic","petrogypsic","placic",
  #                       "salic","spodic","sulfuric"),diags,id)
  #
  # b <- diag_bot_depth(c("duripan","fragipan","argillic","calcic","gypsic",
  #
"natric","oxic","petrocalcic","petrogypsic","placic",
  #                       "salic","spodic","sulfuric"),diags,id)
  #
  # epi_top <- diags[["epipedon"]]["top_depth"][diags[["epipedon"]]
["pedon"]==id,]
  # epi_bot <- diags[["epipedon"]]["bottom_depth"][diags[["epipedon"]]
["pedon"]==id,]
  #
  # epi_sl <- prof_slice(epi_bot,epi_top,hz)
  #
  #
  # if(any(is.na(t))) {
  #
  #   ## fix this in context
  #
  #   sl <- prof_slice(max(hz$hzdepb),epi_bot,hz)
  #
  #   res <- cambic_conds(sl,epi_sl,diags)
  #
  #   diags <- res
  #
  # } else {
  #
  #   ## this is dumb but it works
  #   df1=data.frame(Group=c("epipedon",names(t)),
  #                  Start=sort(c(epi_top,t)),
  #                  End=sort(c(epi_bot,b)))
  #
  #   tops <- vector()
  #   bots <- vector()
  #   for(k in 2:nrow(df1)) {
  #
  #     if(df1$Start[k] != df1$End[k-1]) {
  #
  #       if((df1$Start[k] - df1$End[k-1]) >= 15) {
  #
  #         tops <- append(tops,df1$End[k-1])
  #         bots <- append(bots,df1$Start[k])
  #
  #       }
  #
  #     }
  #   }
  #
  #
```

```
  #   ## if no "empty" space to check for cambic
  #   if(length(tops)==0 & length(bots)==0) {
  #
  #     diags[["cambic"]]["cond3"][diags[["cambic"]]["pedon"]==id,] <- FALSE
  #     diags[["cambic"]]["hz"][diags[["cambic"]]["pedon"]==id,] <- FALSE
  #
  #     return(diags)
  #
  #   }
  #
  #
  #
  #   cambic_dfs <- list()
  #   for(i in length(tops)) {
  #
  #     sl <- prof_slice(bots[i],tops[i],hz)
  #
  #     res <- cambic_conds(sl,epi_sl,diags)
  #     res_df <- res[["cambic"]][res[["cambic"]]["pedon"]==id,]
  #
  #     cambic_dfs <- append(cambic_dfs, list(res_df))
  #
  #
  #   }
  #
  #   ## look through dfs in the list and find if any are true
  #   ## 1. if one is true, keep it
  #   ## 2. if none are true, keep one that had most true conds
  #   ## 3. if more than one is true, throw BIG warning with depth ranges;
keep thickest
  #
  #   l <- unlist(lapply(cambic_dfs,function(x){x[["hz"]]}))
  #   if(sum(l)==1) {
  #
  #     diags[["cambic"]][diags[["cambic"]]["pedon"]==id,] <-
as.data.frame(cambic_dfs[l])
  #
  #   }
  #
  #   if(sum(l)==0) {
  #
  #     lt <- unlist(lapply(cambic_dfs,function(x)
{length(which(as.data.frame(x)==TRUE))}))
  #     diags[["cambic"]][diags[["cambic"]]["pedon"]==id,] <-
as.data.frame(cambic_dfs[which.max(lt)])
  #
  #   }
  #
  #   if(sum(l)>1) {
  #
  #     warning("Cambic horizon found in multiple parts of the same
pedon.Ensure input data is correct.")
  #
  #     lthk <- unlist(lapply(cambic_dfs,function(x){x[["thick"]]}))
```

```
  #     diags[["cambic"]][diags[["cambic"]]["pedon"]==id,] <-
as.data.frame(cambic_dfs[which.max(lthk)])
  #
  #   }
  #
  # }
  #
  #
  return(diags)



}
```

```
#############################################################################
# just check if cemented and ask for slake information
# - what if slake is exactly 50%?
# - consider removing slake parameters and replace with hz designations
duripan <- function(hzs, diags, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)

  cond1 <- which(hz$cemented)

  if(length(cond1)==0) {

    diags[["duripan"]]["cond1"][diags[["duripan"]]["pedon"]==id,] <- FALSE
    diags[["duripan"]]["hz"][diags[["duripan"]]["pedon"]==id,] <- FALSE
    return(diags)

  }


  ## not checking condition 2 ##

  # volume that slakes during a slake test
  cond3 <- cond1[cond1 %in% which((hz$slake_hcl < 50 | hz$slake_h2o < 50) &
                                  (hz$slake_koh > 50 | hz$slake_naoh > 50))]


  if(length(cond3)!=0) {

    diags[["duripan"]]["cond3"][diags[["duripan"]]["pedon"]==id,] <- TRUE
    diags[["duripan"]]["hz"][diags[["duripan"]]["pedon"]==id,] <- TRUE
```

```
    diags[["duripan"]]["top_depth"][diags[["duripan"]]["pedon"]==id,] <-
min(hz$hzdept[cond3])
    diags[["duripan"]]["bottom_depth"][diags[["duripan"]]["pedon"]==id,] <-
max(hz$hzdepb[cond3])
    diags[["duripan"]]["thick"][diags[["duripan"]]["pedon"]==id,] <-
max(hz$hzdepb[cond3]) - min(hz$hzdept[cond3])
    return(diags)

  }

  diags[["duripan"]]["cond3"][diags[["duripan"]]["pedon"]==id,] <- FALSE
  diags[["duripan"]]["hz"][diags[["duripan"]]["pedon"]==id,] <- FALSE
  return(diags)

}









################################################################################
# what can we check for "evidence of pedogenesis"?
# - checks for "x" until a better assumption for manner of failure
fragipan <- function(hzs, diags, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)

  cond0 <- grep(hz$hzname,pattern="[x]")

  if(length(cond0)==0) {

    if(all(is.na(hz$failure))) {

      diags[["fragipan"]]["cond0"][diags[["fragipan"]]["pedon"]==id,] <- FALSE
      diags[["fragipan"]]["hz"][diags[["fragipan"]]["pedon"]==id,] <- FALSE
      return(diags)

    }

  } else {

    diags[["fragipan"]]["cond0"][diags[["fragipan"]]["pedon"]==id,] <- TRUE
    diags[["fragipan"]]["hz"][diags[["fragipan"]]["pedon"]==id,] <- TRUE
    diags[["fragipan"]]["top_depth"][diags[["fragipan"]]["pedon"]==id,] <-
min(hz$hzdept[cond0])
    diags[["fragipan"]]["bottom_depth"][diags[["fragipan"]]["pedon"]==id,] <-
max(hz$hzdepb[cond0])
```

```
    diags[["fragipan"]]["thick"][diags[["fragipan"]]["pedon"]==id,] <-
max(hz$hzdepb[cond0]) - min(hz$hzdept[cond0])
    return(diags)

  }



  cond3 <- which(tolower(hz$structtype) %in%
c("prismatic","columnar","subangular blocky","angular blocky","massive") |
                 tolower(hz$structgrade) == "weak")

  if(length(cond3)==0) {

    diags[["fragipan"]]["cond3"][diags[["fragipan"]]["pedon"]==id,] <- FALSE
    diags[["fragipan"]]["hz"][diags[["fragipan"]]["pedon"]==id,] <- FALSE
    return(diags)

  }



  cond4 <- cond3[cond3 %in% which(hz$slake_h2o > 50)]

  if(length(cond4)==0) {

    diags[["fragipan"]]["cond4"][diags[["fragipan"]]["pedon"]==id,] <- FALSE
    diags[["fragipan"]]["hz"][diags[["fragipan"]]["pedon"]==id,] <- FALSE
    return(diags)

  }


  ## roots not checked
  cond5 <- cond4[cond4 %in%
                 which(tolower(hz$rrc) %in% c("firm","very firm","extremely
firm","slightly rigid","rigid","very rigid") &
                       tolower(hz$failure) == "brittle")]


  if(length(cond5)==0) {

    diags[["fragipan"]]["cond5"][diags[["fragipan"]]["pedon"]==id,] <- FALSE
    diags[["fragipan"]]["hz"][diags[["fragipan"]]["pedon"]==id,] <- FALSE
    return(diags)

  }



  cond6 <- cond5[cond5 %in% which(tolower(hz$effclass) == "noneffervescent")]
```

```
  if(length(cond6)==0) {

    diags[["fragipan"]]["cond6"][diags[["fragipan"]]["pedon"]==id,] <- FALSE
    diags[["fragipan"]]["hz"][diags[["fragipan"]]["pedon"]==id,] <- FALSE
    return(diags)

  }



  if(sum(hz$hzthk[cond6]) >= 15) {

    diags[["fragipan"]]["cond1"][diags[["fragipan"]]["pedon"]==id,] <- TRUE
    diags[["fragipan"]]["hz"][diags[["fragipan"]]["pedon"]==id,] <- TRUE
    diags[["fragipan"]]["top_depth"][diags[["fragipan"]]["pedon"]==id,] <-
min(hz$hzdept[cond6])
    diags[["fragipan"]]["bottom_depth"][diags[["fragipan"]]["pedon"]==id,] <-
max(hz$hzdepb[cond6])
    diags[["fragipan"]]["thick"][diags[["fragipan"]]["pedon"]==id,] <-
max(hz$hzdepb[cond6]) - min(hz$hzdept[cond6])
    return(diags)

  }


  diags[["fragipan"]]["cond1"][diags[["fragipan"]]["pedon"]==id,] <- FALSE
  diags[["fragipan"]]["hz"][diags[["fragipan"]]["pedon"]==id,] <- FALSE
  return(diags)


}









################################################################################
# SKIP FOR NOW
# - define eluvial part; add dependency on albic?
# - define illuvial part
glossic <- function(hzs, diags, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)

  return(diags)

}
```

```
#############################################################################
# TODO:
# - if 'y', assumes at least 5%
# - if 'yy', assumes at least 50%
gypsic <- function(hzs, diags, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)


  cond2 <- which(!hz$cemented)


  if(length(cond2)==0) {

    diags[["gypsic"]]["cond2"][diags[["gypsic"]]["pedon"]==id,] <- FALSE
    diags[["gypsic"]]["hz"][diags[["gypsic"]]["pedon"]==id,] <- FALSE
    return(diags)

  }



  # test for secondary gypsum?
  cond3 <- cond2[cond2 %in% which(hz$gypsum >= 5)]

  if(length(cond3)==0) {

    diags[["gypsic"]]["cond3"][diags[["gypsic"]]["pedon"]==id,] <- FALSE
    diags[["gypsic"]]["hz"][diags[["gypsic"]]["pedon"]==id,] <- FALSE
    return(diags)

  }




  thk <- sum(hz$hzthk[cond3])

  if(thk < 15) {

    diags[["gypsic"]]["cond1"][diags[["gypsic"]]["pedon"]==id,] <- FALSE
    diags[["gypsic"]]["hz"][diags[["gypsic"]]["pedon"]==id,] <- FALSE
    return(diags)
```

```
  }

  diags[["gypsic"]]["cond1"][diags[["gypsic"]]["pedon"]==id,] <- TRUE



  gypsum_wa <- weighted.mean(hz$gypsum[cond3],hz$hzthk[cond3]/thk)
  cond4 <- thk*gypsum_wa

  if(cond4 >= 150){

    diags[["gypsic"]]["cond4"][diags[["gypsic"]]["pedon"]==id,] <- TRUE
    diags[["gypsic"]]["hz"][diags[["gypsic"]]["pedon"]==id,] <- TRUE
    diags[["gypsic"]]["top_depth"][diags[["gypsic"]]["pedon"]==id,] <-
min(hz$hzdept[cond3])
    diags[["gypsic"]]["bottom_depth"][diags[["gypsic"]]["pedon"]==id,] <-
max(hz$hzdepb[cond3])
    diags[["gypsic"]]["thick"][diags[["gypsic"]]["pedon"]==id,] <-
max(hz$hzdepb[cond3]) - min(hz$hzdept[cond3])
    return(diags)

  }

  diags[["gypsic"]]["cond4"][diags[["gypsic"]]["pedon"]==id,] <- FALSE
  diags[["gypsic"]]["hz"][diags[["gypsic"]]["pedon"]==id,] <- FALSE
  return(diags)


}









################################################################################
# TODO:
# - not checking cond 6
# - currently checking lower bound as last t horizon
kandic <- function(hzs, diags, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)



  minsurf <- dplyr::filter(diags[["orgmin"]],pedon==id)[["min_top"]]
  has_t <- grepl(as.character(hz$hzname), pattern = "[Tt]")

  ## simplified; if no t, there is no kandic
```

```
  ## call it cond1 for now
  if(!any(has_t)) {

    diags[["kandic"]]["cond1"][diags[["kandic"]]["pedon"]==id,] <- FALSE
    diags[["kandic"]]["hz"][diags[["kandic"]]["pedon"]==id,] <- FALSE

    return(diags)
  }


  # check if contact within 50cm of surface
  cond1_hz <- c("densic","lithic","paralithic","petroferric")
  cond1_contact <- any(lapply(diags[cond1_hz],function(x){x[["top_depth"]]
[which(x[["pedon"]]==id)]})<=50)

  if(!is.na(cond1_contact) & cond1_contact) {

    cond1_contact_depth <- min(as.numeric(
                               lapply(diags[cond1_hz],function(x)
{x[["top_depth"]][which(x[["pedon"]]==id)]})),na.rm=TRUE)

  }




  ## Find upper bound ##
  upper.bound <- kandic.clay.increase.depth(hz,"claytotal")


  if(is.na(upper.bound)) {

    diags[["kandic"]]["cond2a"][diags[["kandic"]]["pedon"]==id,] <- FALSE
    diags[["kandic"]]["hz"][diags[["kandic"]]["pedon"]==id,] <- FALSE
    return(diags)

  }

  ## which hz is upper bound in ##
  upper.bound.hz <- max(which(hz$hzdept <= upper.bound & hz$hzdepb >=
upper.bound))
  overlyhz <- upper.bound.hz-1


  ## Check cond 2b1 ##
  top_100 <- prof_slice(bottom=minsurf+100,top=minsurf,hz=hz)
  cond2b1_textures <- c("coarse sand","sand","fine sand","loamy coarse
sand","loamy sand","loamy fine sand")
  cond2b1 <- all(tolower(top_100$texture) %in% c(cond2b1_textures))

  if(cond2b1 & (upper.bound >=minsurf+100 & upper.bound <=minsurf+200)) {

    diags[["kandic"]]["cond2b"][diags[["kandic"]]["pedon"]==id,] <- TRUE
```

```
    diags[["kandic"]]["cond2b1"][diags[["kandic"]]["pedon"]==id,] <- TRUE


  } else if(all(hz$claytotal[overlyhz]>=20) & (upper.bound <= minsurf+100)) {

    diags[["kandic"]]["cond2b"][diags[["kandic"]]["pedon"]==id,] <- TRUE
    diags[["kandic"]]["cond2b1"][diags[["kandic"]]["pedon"]==id,] <- FALSE
    diags[["kandic"]]["cond2b2"][diags[["kandic"]]["pedon"]==id,] <- TRUE

  } else if(upper.bound <= minsurf+125) {

    diags[["kandic"]]["cond2b"][diags[["kandic"]]["pedon"]==id,] <- TRUE
    diags[["kandic"]]["cond2b1"][diags[["kandic"]]["pedon"]==id,] <- FALSE
    diags[["kandic"]]["cond2b2"][diags[["kandic"]]["pedon"]==id,] <- FALSE
    diags[["kandic"]]["cond2b3"][diags[["kandic"]]["pedon"]==id,] <- TRUE

  } else {

    diags[["kandic"]]["cond2b"][diags[["kandic"]]["pedon"]==id,] <- FALSE
    diags[["kandic"]]["cond2b1"][diags[["kandic"]]["pedon"]==id,] <- FALSE
    diags[["kandic"]]["cond2b2"][diags[["kandic"]]["pedon"]==id,] <- FALSE
    diags[["kandic"]]["cond2b3"][diags[["kandic"]]["pedon"]==id,] <- FALSE
    diags[["kandic"]]["hz"][diags[["kandic"]]["pedon"]==id,] <- FALSE
    return(diags)

  }


  ## continue if one evaluates to true ##

  ## simplified condition 5 ##
  # inital lower bound
  lower.bound <- hz$hzdepb[max(has_t)]
  cond5_slice <- prof_slice(bottom=lower.bound,top=upper.bound,hz=hz)
  cond5_idx <- which(cond5_slice$acec7 <= 16 & cond5_slice$aecec <= 12)
  cond5_split <- split(cond5_idx, cumsum(c(1, diff(cond5_idx) != 1)))

  # get first sequence of continuous that also contains the upper bound
  cond5 <- as.numeric(which(lapply(cond5_split,function(x){upper.bound.hz %in%
x})==TRUE))


  if(length(cond5)==0){

    diags[["kandic"]]["cond5"][diags[["kandic"]]["pedon"]==id,] <- FALSE
    diags[["kandic"]]["hz"][diags[["kandic"]]["pedon"]==id,] <- FALSE
    return(diags)

  } else {

    lower.bound <- max(cond5_slice$hzdepb[cond5])

  }
```

```
  ## condition 4 ##

  cond4_slice <- prof_slice(bottom=lower.bound,top=upper.bound,hz=hz)
  cond4_textures <- c("loamy very find sand","coarse sandy loam","sandy
loam","fine sandy loam",
                      "very fine sandy loam","loam","silt loam","silt","sandy
clay loam",
                      "clay loam","silty clay loam","sandy clay","silty
clay","clay")
  cond4 <- all(cond4_slice$texture %in% cond4_textures)

  if(!cond4) {

    diags[["kandic"]]["cond4"][diags[["kandic"]]["pedon"]==id,] <- FALSE
    diags[["kandic"]]["hz"][diags[["kandic"]]["pedon"]==id,] <- FALSE
    return(diags)

  } else {

    diags[["kandic"]]["cond4"][diags[["kandic"]]["pedon"]==id,] <- TRUE

  }



  ## check overlaying condition 1 ##
  if(hz$hzthk[overlyhz] >= 18) {

    diags[["kandic"]]["cond1"][diags[["kandic"]]["pedon"]==id,] <- TRUE

  } else if(hz$hzthk[overlyhz] >= 5 & !cond1_contact) {

    diags[["kandic"]]["cond1"][diags[["kandic"]]["pedon"]==id,] <- TRUE

  } else {

    diags[["kandic"]]["cond1"][diags[["kandic"]]["pedon"]==id,] <- FALSE
    diags[["kandic"]]["hz"][diags[["kandic"]]["pedon"]==id,] <- FALSE

    return(diags)

  }



  ## check thickness ##
```

```
  kandic_thk <- upper.bound - lower.bound


  if(kandic_thk >= 30) {

    diags[["kandic"]]["cond3a"][diags[["kandic"]]["pedon"]==id,] <- TRUE
    diags[["kandic"]]["hz"][diags[["kandic"]]["pedon"]==id,] <- TRUE
    diags[["kandic"]]["top_depth"][diags[["kandic"]]["pedon"]==id,] <-
upper.bound
    diags[["kandic"]]["bottom_depth"][diags[["kandic"]]["pedon"]==id,] <-
lower.bound
    diags[["kandic"]]["thick"][diags[["kandic"]]["pedon"]==id,] <- lower.bound
- upper.bound
    return(diags)

  } else if(cond1_contact & (kandic_thk / (cond1_contact_depth-18))>=0.6 &
kandic_thk>=15) {

    diags[["kandic"]]["cond3b"][diags[["kandic"]]["pedon"]==id,] <- TRUE
    diags[["kandic"]]["hz"][diags[["kandic"]]["pedon"]==id,] <- TRUE
    diags[["kandic"]]["top_depth"][diags[["kandic"]]["pedon"]==id,] <-
upper.bound
    diags[["kandic"]]["bottom_depth"][diags[["kandic"]]["pedon"]==id,] <-
lower.bound
    diags[["kandic"]]["thick"][diags[["kandic"]]["pedon"]==id,] <- lower.bound
- upper.bound
    return(diags)

  } else {

    diags[["kandic"]]["cond3a"][diags[["kandic"]]["pedon"]==id,] <- FALSE
    diags[["kandic"]]["cond3b"][diags[["kandic"]]["pedon"]==id,] <- FALSE
    diags[["kandic"]]["hz"][diags[["kandic"]]["pedon"]==id,] <- FALSE
    return(diags)

  }


  return(diags)
  ## not checking condition 6 right now ##



}
```

```
################################################################################
# - copied Argillic with added conditions
# - check that condition codes line up
natric <- function(hzs, diags, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)


  # Gets indices of horizons with "t" subordinate designation
  # Used to generally satisfy all components of cond1b
  cond1b <- grep(hz$hzname,pattern="[Tt]")

  if(length(cond1b)!=0) {
    diags[["natric"]]["cond1b"][diags[["natric"]]["pedon"]==id,] <- TRUE
  }


  ## The following is adapted from AQP ##

  hzdesgn <- 'hzname'
  require_t <- TRUE
  bottom.pattern <- "Cr|R|Cd"
  lower.grad.pattern <- "^[2-9]*B*CB*[^rtd]*[1-9]*$"
  # sandy.texture.pattern <- "-S$|^S$|COS$|L[^V]FS$|[^L]VFS$|LS$|LFS$"
  sandy.texture.pattern <- c("sand","coarse sand","loamy coarse sand",
                             "loamy fine sand","loamy sand")
  verbose <- FALSE

  depthcol <- c("hzdept","hzdepb")


  has_t <- grepl(as.character(hz$hzname), pattern = "[Tt]")
  pld <- hz$hzdepb[grep(hz$hzname,pattern="[Tp]")]
  mss <- dplyr::filter(diags[["orgmin"]],pedon==id)[["min_top"]]


  ## simplified; if no t, there is no kandic
  ## call it cond1 for now
  if(!any(has_t)) {

    # diags[["natric"]]["cond1"][diags[["natric"]]["pedon"]==id,] <- FALSE
    diags[["natric"]]["hz"][diags[["natric"]]["pedon"]==id,] <- FALSE

    return(diags)
  }



  if(length(pld)==0) {
    pld <- FALSE
  } else if(length(pld)>1) {
    pld <- pld[length(pld)]
```

```
}

# elluv <- shallowest_t-1 # index of the elluvial hz above the first t
# elluv_clay <- hz$claytotal[elluv]

upper.bound <- argillic.clay.increase.depth(hz,"claytotal")
lower.bound <- -Inf
soil.depth.idx <- (hz$densic | hz$lithic | hz$paralithic)
if(any(soil.depth.idx)) {
  soil.depth <- min(hz$hzdept[soil.depth.idx])
} else {
  soil.depth <- max(hz$hzdepb)
}



## ensure upper bound is in t hz
if(length(upper.bound)!=0 & !is.na(upper.bound)) {

  if(!grepl(hz$hzname[which(hz$hzdept == upper.bound)],pattern="[Tt]")) {

    if(length(cond1b)!=0) {
      shallowest_t <- min(hz$hzdept[cond1b]) # index of the first t
      upper.bound <- shallowest_t
    } else {
      shallowest_t <- NA
    }
  }
}



if(is.na(upper.bound)) {

  if(length(cond1b)!=0) {
    shallowest_t <- min(hz$hzdept[cond1b]) # index of the first t
  } else {
    shallowest_t <- NA
  }

  depth_ld <- hz$hzdept[grepl(hz$hzname, pattern = "^[2-9].*[Tt]")]

  if (!is.na(shallowest_t)) {
    if (pld == shallowest_t) {
      upper.bound <- shallowest_t
    }

    if (!all(is.na(depth_ld))) {
      if (any(shallowest_t %in% depth_ld)) {
        upper.bound <- shallowest_t
      }
    }
  }
}
```

```
  ## Section adapted from AQP ##

  if (!is.na(upper.bound)) {

    ########
    # TODO: allow evidence of illuviation from lab data fine clay ratios etc?
how?
    #
    # TODO: if `require_t` is set to `FALSE`... or in a future
getKandicBounds()...
    #       how do you detect the bottom of a argillic or kandic horizon
(which need not have clay films)
    #          in e.g. saprolite ?
    #       could you look for C in master horizon designation for e.g. rock
structure / parent material?
    #       in lieu of checking for t and a cemented bedrock contact... is
that how lower.bound should be determined?
    ########
    if (sum(has_t) | !require_t) {
      # get index of last horizon with t
      idx.last <- rev(which(has_t))[1]

      ## Partial fix for TODO #2? seems reasnable for the require_t=FALSE
lower.bound case
      # take _very_ last horizon depth first (will be truncated to contact
depth if needed)
      depth.last <- as.numeric(aqp:::.data.frame.j(hz[nrow(hz),], depthcol[2],
"data.frame"))

      # take the top depth of any B or C horizon  without t subscript above
"depth.last"
      c.idx <- which(grepl(hz[[hzdesgn]], pattern = lower.grad.pattern))
      if (length(c.idx)) {
        c.horizon <- as.numeric(aqp:::.data.frame.j(hz[c.idx[1], ],
depthcol[1], "data.frame"))

        # if the _shallowest C horizon_ top depth is above the _last horizon_
bottom depth (could be top depth of same hz)
        if (c.horizon < depth.last)  {
          # use the top depth of the first C horizon that matched the pattern
          if (verbose) {
            message(paste0("Found ",paste0(hz[[hzdesgn]][c.idx], collapse =
","),
                           " below natric, adjusting lower bound (",
                           "pedon_id",": ", as.character(id),")"))
            # plot(p)
            # print(c.idx)
            depth.last <- c.horizon
          }
        }
      }
```

```
      # get the bottom depth of the last horizon with a t (this could be same
as c.horizon above)
      if (require_t) {
        depth.last <- as.numeric(aqp:::.data.frame.j(hz[idx.last,],
                                                     depthcol[2],
                                                     "data.frame"))
      }

      # in rare cases, the bottom depth of the bedrock/contact is not
populated
      # step back until we find one that is not NA
      idx.last.i <- idx.last
      while (is.na(depth.last) & idx.last.i >= 0) {
        depth.last <- as.numeric(aqp:::.data.frame.j(hz[idx.last.i,],
                                                     depthcol[2],
                                                     "data.frame"))
        idx.last.i <- idx.last.i - 1
      }

      # if the last horizon with a t is below the contact (Crt or Rt) or some
other weird reason
      if (soil.depth < depth.last) {
        # return the soil depth to contact
        lower.bound <- soil.depth

      } else {
        #otherwise, return the bottom depth of the last horizon with a t
        lower.bound <- depth.last
      }
    } else {
      if (verbose) {
        message(paste0("Profile (",as.character(id),") has clay increase with
no evidence of illuviation (t)."))
        lower.bound <- NA
        upper.bound <- NA
      }
    }
  } else {

    # if the upper bound is NA, return NA for the lower bound
    lower.bound <- NA
  }

  if (!is.finite(lower.bound)) {
    lower.bound <- NA
  }

  if (is.na(upper.bound)) {
    #return(c(ubound = NA, lbound = NA))
    diags[["natric"]]["hz"][diags[["natric"]]["pedon"]==id,] <- FALSE
  }
```

```
  if(!is.na(upper.bound)) {
    #bdepthspc <- glom(p, mss, upper.bound, df = TRUE)
    bdepthspc <- glom_br(hz=hz, pedon_id=id, top=mss, bottom=upper.bound)
  } else {
    bdepthspc <- NA
  }


  # if there are no overlying horizons, return NA
  if (is.null(bdepthspc) | all(is.na(bdepthspc))) {
    if (verbose) {
      message(paste0("Profile (",id,
                     ") has no horizons overlying a [possible] natric."))

    }
    #return(c(ubound = NA, lbound = NA))
    diags[["natric"]]["hz"][diags[["natric"]]["pedon"]==id,] <- FALSE
  }


  if (all(!is.na(c(upper.bound, lower.bound)))) {
    bdepths <- bdepthspc[[depthcol[2]]]

    # if argi bounds are found check that minimum thickness requirements are
met
    min.thickness <- max(7.5, max(bdepths, na.rm = TRUE) / 10)

    #textures <- glom(p, upper.bound, lower.bound, df = TRUE)[[texcl.attr]]
    textures <- glom_br(hz=hz, top=upper.bound, bottom=lower.bound)
[["texture"]]
    # is_sandy <- all(grepl(sandy.texture.pattern, textures, ignore.case =
TRUE))
    is_sandy <- all(textures %in% sandy.texture.pattern)


    if (is_sandy) {
      min.thickness <- 15
    }

    if (lower.bound - upper.bound < min.thickness) {
      if (verbose) {
        message(paste0("Profile (",id,
                       ") does not meet thickness requirement."))
        #return(c(ubound = NA, lbound = NA))

      }
      diags[["natric"]]["hz"][diags[["natric"]]["pedon"]==id,] <- FALSE
      diags[["natric"]]["cond1a"][diags[["natric"]]["pedon"]==id,] <- FALSE
    }
  }


  # this is redundant; need to rearrange
```

```
  if(is.na(upper.bound) & is.na(upper.bound)) {

    diags[["natric"]]["hz"][diags[["natric"]]["pedon"]==id,] <- FALSE
    return(diags)

  }



  ##### Specific to Natric #####

  ## Cond 4


  cond4_slice <- prof_slice(bottom=lower.bound,top=upper.bound,hz=hz)
  cond4a <- which(cond4_slice$structtype %in% c("columnar","prismatic"))


  if(length(cond4a)>=1) {

    diags[["natric"]]["cond4a"][diags[["natric"]]["pedon"]==id,] <- TRUE

  } else {

    diags[["natric"]]["cond4a"][diags[["natric"]]["pedon"]==id,] <- FALSE
    diags[["natric"]]["hz"][diags[["natric"]]["pedon"]==id,] <- FALSE
    return(diags)

  }

  ## cond 4b not checked ##



  ## Cond 5
  cond5a_slice <- prof_slice(bottom=upper.bound+40,top=upper.bound,hz=hz)
  cond5a <- which(cond5a_slice$esp >= 15 | cond5a_slice$sar >= 13)

  minsurf <- dplyr::filter(diags[["orgmin"]],pedon==id)[["min_top"]]

  cond5b_slice1 <- prof_slice(bottom=upper.bound+40,top=upper.bound,hz=hz)
  cond5b_slice2 <- prof_slice(bottom=minsurf+200,top=minsurf,hz=hz)

  cond5b_1 <- which((cond5b_slice1$ex_mg+cond5b_slice1$ex_na) >
(cond5b_slice1$ex_ca+cond5b_slice1$extracid))
  cond5b_2 <- which(cond5b_slice2$esp >= 15 | cond5b_slice2$sar >= 13)


  if(length(cond5a)>=1) {

    diags[["natric"]]["cond5a"][diags[["natric"]]["pedon"]==id,] <- TRUE
```

```
  } else if(length(cond5b_1)>=1 | length(cond5b_2) >=1) {

    diags[["natric"]]["cond5a"][diags[["natric"]]["pedon"]==id,] <- FALSE
    diags[["natric"]]["cond5b"][diags[["natric"]]["pedon"]==id,] <- TRUE

  } else {

    diags[["natric"]]["cond5a"][diags[["natric"]]["pedon"]==id,] <- FALSE
    diags[["natric"]]["cond5b"][diags[["natric"]]["pedon"]==id,] <- FALSE
    return(diags)


  }
```

```
  if (!is.na(upper.bound)) {
    # it is possible that a subhorizon of the Ap horizon meets the clay
increase
    if (pld > upper.bound) {
      upper.bound <- pld
      if (verbose) {
        message(paste0("Profile (",id,
                       ") meets clay increase within plowed layer."))

        diags[["natric"]]["cond2"][diags[["natric"]]["pedon"]==id,] <- TRUE
        diags[["natric"]]["hz"][diags[["natric"]]["pedon"]==id,] <- TRUE
        diags[["natric"]]["top_depth"][diags[["natric"]]["pedon"]==id,] <-
upper.bound
        diags[["natric"]]["bottom_depth"][diags[["natric"]]["pedon"]==id,] <-
lower.bound
        diags[["natric"]]["thick"][diags[["natric"]]["pedon"]==id,] <-
lower.bound - upper.bound
      }

    }
  }
  #return(c(ubound = upper.bound, lbound = lower.bound))
  return(diags)

}
```

```
#############################################################################
ortstein <- function(hzs, diags, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)


  # spodic_mat <- as.logical(lapply(diags["spodic_mat"],function(x)
{x[["prop"]][which(x[["pedon"]]==id)]}))
  spodic_mat <- diags[["spodic_mat"]]["prop"][diags[["spodic_mat"]]
["pedon"]==id,]

  if(!spodic_mat) {

    diags[["ortstein"]]["cond1"][diags[["ortstein"]]["pedon"]==id,] <- FALSE
    diags[["ortstein"]]["hz"][diags[["ortstein"]]["pedon"]==id,] <- FALSE
    return(diags)

  }


  spodic_top <- diags[["spodic_mat"]]["top_depth"][diags[["spodic_mat"]]
["pedon"]==id,]
  spodic_bot <- diags[["spodic_mat"]]["bottom_depth"][diags[["spodic_mat"]]
["pedon"]==id,]

  cond1 <- which(hz$hzdept >= spodic_top & hz$hzdepb <= spodic_bot)
  cond2 <- cond1[cond1 %in% which(hz$cemented)]
  cond3 <- cond2[cond2 %in% which(hz$hzthk >= 2.5)]

  if(length(cond2)==0) {

    diags[["ortstein"]]["cond2"][diags[["ortstein"]]["pedon"]==id,] <- FALSE
    diags[["ortstein"]]["hz"][diags[["ortstein"]]["pedon"]==id,] <- FALSE

  } else {

    diags[["ortstein"]]["cond2"][diags[["ortstein"]]["pedon"]==id,] <- TRUE

  }


  if(length(cond3)==0) {

    diags[["ortstein"]]["cond3"][diags[["ortstein"]]["pedon"]==id,] <- FALSE
    diags[["ortstein"]]["hz"][diags[["ortstein"]]["pedon"]==id,] <- FALSE
    return(diags)

  }
```

```
  diags[["ortstein"]]["cond3"][diags[["ortstein"]]["pedon"]==id,] <- TRUE
  diags[["ortstein"]]["hz"][diags[["ortstein"]]["pedon"]==id,] <- TRUE
  diags[["ortstein"]]["top_depth"][diags[["ortstein"]]["pedon"]==id,] <-
min(hz$hzdept[cond3])
  diags[["ortstein"]]["bottom_depth"][diags[["ortstein"]]["pedon"]==id,] <-
max(hz$hzdepb[cond3])
  diags[["ortstein"]]["thick"][diags[["ortstein"]]["pedon"]==id,] <-
max(hz$hzdepb[cond3]) - min(hz$hzdept[cond3])
  return(diags)


}









################################################################################
# TODO:
# - find lower bound
# - consider 'o' subordinate
# - add cond 3 and cond 4
oxic <- function(hzs, diags, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)



  ## cond 0: andic properties
  cond0 <- as.logical(lapply(diags["andic_prop"],function(x){x[["prop"]]
[which(x[["pedon"]]==id)]}))

  ## o subordinates
  cond0_2 <- grep(hz$hzname,pattern="o")

  if(cond0) {

    diags[["oxic"]]["cond0"][diags[["oxic"]]["pedon"]==id,] <- FALSE
    diags[["oxic"]]["hz"][diags[["oxic"]]["pedon"]==id,] <- FALSE
    return(diags)

  }

  diags[["oxic"]]["cond0"][diags[["oxic"]]["pedon"]==id,] <- TRUE
```

```
  ## cond 2: texture class
  cond2_textures <- c("sandy loam","fine sandy loam","very fine sandy
loam","loamd",
                      "silt loam","silt","sandy clay loam","clay loam","silty
clay loam",
                      "sandy clay","silty clay","clay")
  # cond2 <- which(hz$texture %in% cond2_textures)
  cond2 <- cond0_2[cond0_2 %in% which(hz$texture %in% cond2_textures)]

  if(length(cond2)==0) {

    diags[["oxic"]]["cond2"][diags[["oxic"]]["pedon"]==id,] <- FALSE
    diags[["oxic"]]["hz"][diags[["oxic"]]["pedon"]==id,] <- FALSE
    return(diags)

  }

  diags[["oxic"]]["cond2"][diags[["oxic"]]["pedon"]==id,] <- TRUE



  ## cond 3 not checked ##


  ## cond 4 not checked ##




  ## cond 5: clay increase

  cond5_sl <- prof_slice(max(hz$hzdepb[cond2]),min(hz$hzdept[cond2]-1),hz)
  upper.bound <- oxic.clay.increase.depth(cond5_sl,"claytotal")

  if(is.na(upper.bound)) {

    diags[["oxic"]]["cond5"][diags[["oxic"]]["pedon"]==id,] <- FALSE
    diags[["oxic"]]["hz"][diags[["oxic"]]["pedon"]==id,] <- FALSE
    return(diags)

  }

  diags[["oxic"]]["cond5"][diags[["oxic"]]["pedon"]==id,] <- TRUE
  upper.bound.hz <- which(hz$hzdept <= upper.bound & hz$hzdepb >= upper.bound)
  if(length(upper.bound.hz)==2){
    upper.bound.hz <- max(upper.bound.hz)
  }
```

```
  ## cond 6: CEC

  cond6_idx <- cond2[cond2 %in% which(hz$acec7 <= 16 & hz$aecec <= 12)]
  cond6_split <- split(cond6_idx, cumsum(c(1, diff(cond6_idx) != 1)))

  cond6 <- unlist(cond6_split[as.numeric(which(lapply(cond6_split,function(x)
{upper.bound.hz %in% x})==TRUE))])

  if(length(cond6)==0) {

    diags[["oxic"]]["cond6"][diags[["oxic"]]["pedon"]==id,] <- FALSE
    diags[["oxic"]]["hz"][diags[["oxic"]]["pedon"]==id,] <- FALSE
    return(diags)

  }

  diags[["oxic"]]["cond6"][diags[["oxic"]]["pedon"]==id,] <- TRUE

  hz_slice <- prof_slice(bottom=max(hz$hzdepb[cond6]),top=upper.bound,hz=hz)
  hz_slice$hzthk <- hz_slice$hzdepb - hz_slice$hzdept




  ## cond 1: thickness

  if(sum(hz_slice$hzthk)<30) {

    diags[["oxic"]]["cond1"][diags[["oxic"]]["pedon"]==id,] <- FALSE
    diags[["oxic"]]["hz"][diags[["oxic"]]["pedon"]==id,] <- FALSE
    return(diags)

  }



  diags[["oxic"]]["hz"][diags[["oxic"]]["pedon"]==id,] <- TRUE
  diags[["oxic"]]["top_depth"][diags[["oxic"]]["pedon"]==id,] <-
min(hz_slice$hzdept)
  diags[["oxic"]]["bottom_depth"][diags[["oxic"]]["pedon"]==id,] <-
max(hz_slice$hzdepb)
  diags[["oxic"]]["thick"][diags[["oxic"]]["pedon"]==id,] <-
max(hz_slice$hzdepb) - min(hz_slice$hzdept)
  return(diags)



}
```

```
#############################################################################
# TODO:
# - add cond 2
# - how to check cond 3b?
# - revise cemagt
# - kk
petrocalcic <- function(hzs, diags, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)

  # cementing agent: primary cementing component >50%
  cond1 <- which(hz$cemented & (hz$cemagt=="carbonates" |
hz$cemagt=="carbonates/silica"))

  ## cond 2 not checked ##

  if(length(cond1)==0) {

    diags[["petrocalcic"]]["cond1"][diags[["petrocalcic"]]["pedon"]==id,] <-
FALSE
    diags[["petrocalcic"]]["hz"][diags[["petrocalcic"]]["pedon"]==id,] <-
FALSE
    return(diags)

  }

  diags[["petrocalcic"]]["cond1"][diags[["petrocalcic"]]["pedon"]==id,] <-
TRUE

  if(sum(hz$hzthk[cond1])>=10) {

    diags[["petrocalcic"]]["cond3a"][diags[["petrocalcic"]]["pedon"]==id,] <-
TRUE
    diags[["petrocalcic"]]["hz"][diags[["petrocalcic"]]["pedon"]==id,] <- TRUE
    diags[["petrocalcic"]]["top_depth"][diags[["petrocalcic"]]["pedon"]==id,]
<- min(hz$hzdept[cond1])
    diags[["petrocalcic"]]["bottom_depth"][diags[["petrocalcic"]]
["pedon"]==id,] <- max(hz$hzdepb[cond1])
    diags[["petrocalcic"]]["thick"][diags[["petrocalcic"]]["pedon"]==id,] <-
max(hz$hzdepb[cond1]) - min(hz$hzdept[cond1])
    return(diags)

  }


  diags[["petrocalcic"]]["cond3a"][diags[["petrocalcic"]]["pedon"]==id,] <-
FALSE
```

```
  diags[["petrocalcic"]]["hz"][diags[["petrocalcic"]]["pedon"]==id,] <- FALSE
  return(diags)


}









################################################################################
# TODO:
# - add cond 2
# - revise cemagt
# - yy
petrogypsic <- function(hzs, diags, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)


  # cementing agent: primary cementing component >50%
  cond1 <- which(hz$cemented & hz$cemagt=="gypsum")

  ## cond 2 not checked ##

  if(length(cond1)==0) {

    diags[["petrogypsic"]]["cond1"][diags[["petrogypsic"]]["pedon"]==id,] <-
FALSE
    diags[["petrogypsic"]]["hz"][diags[["petrogypsic"]]["pedon"]==id,] <-
FALSE
    return(diags)

  }

  diags[["petrogypsic"]]["cond1"][diags[["petrogypsic"]]["pedon"]==id,] <-
TRUE



  ## cond 4: gypsum content
  wa_gyp <- weighted.mean(hz$gypsum[cond1],hz$hzthk[cond1]/
sum(hz$hzthk[cond1]))

  if(wa_gyp<40) {

    diags[["petrogypsic"]]["cond4"][diags[["petrogypsic"]]["pedon"]==id,] <-
FALSE
```

```
    diags[["petrogypsic"]]["hz"][diags[["petrogypsic"]]["pedon"]==id,] <-
FALSE
    return(diags)

  }

  diags[["petrogypsic"]]["cond4"][diags[["petrogypsic"]]["pedon"]==id,] <-
TRUE



  ## cond 3: thickness

  if(sum(hz$hzthk[cond1])<0.5) {

    diags[["petrogypsic"]]["cond3"][diags[["petrogypsic"]]["pedon"]==id,] <-
FALSE
    diags[["petrogypsic"]]["hz"][diags[["petrogypsic"]]["pedon"]==id,] <-
FALSE
    return(diags)

  }


  diags[["petrogypsic"]]["cond3"][diags[["petrogypsic"]]["pedon"]==id,] <-
TRUE
  diags[["petrogypsic"]]["hz"][diags[["petrogypsic"]]["pedon"]==id,] <- TRUE
  diags[["petrogypsic"]]["top_depth"][diags[["petrogypsic"]]["pedon"]==id,] <-
min(hz$hzdept[cond1])
  diags[["petrogypsic"]]["bottom_depth"][diags[["petrogypsic"]]["pedon"]==id,]
<- max(hz$hzdepb[cond1])
  diags[["petrogypsic"]]["thick"][diags[["petrogypsic"]]["pedon"]==id,] <-
max(hz$hzdepb[cond1]) - min(hz$hzdept[cond1])
  return(diags)


}









################################################################################
# TODO:
# - cond 2 not checked
# - revise cemagt
# - s
placic <- function(hzs, diags, id) {
```

```
  hz <- dplyr::filter(hzs, pedon_id==id)

  # cemented with iron
  cond1 <- which(hz$cemented & hz$cemagt == "iron" & hz$om > 0)


  if(length(cond1)==0) {

    diags[["placic"]]["cond1"][diags[["placic"]]["pedon"]==id,] <- FALSE
    diags[["placic"]]["hz"][diags[["placic"]]["pedon"]==id,] <- FALSE
    return(diags)

  }


  diags[["placic"]]["cond1"][diags[["placic"]]["pedon"]==id,] <- TRUE



  ## This needs to be reviewed
  # spodic_mat <- as.logical(lapply(diags["spodic_mat"],function(x)
{x[["prop"]][which(x[["pedon"]]==id)]}))
  spodic_mat <- any(hz$spodic_mat_pct > 0)

  if(sum(hz$hzthk[cond1]) >= 0.1) {


    if(spodic_mat) {

      if(sum(hz$hzthk[cond1]) < 2.5) {

        diags[["placic"]]["cond3"][diags[["placic"]]["pedon"]==id,] <- TRUE
        diags[["placic"]]["hz"][diags[["placic"]]["pedon"]==id,] <- TRUE
        diags[["placic"]]["top_depth"][diags[["placic"]]["pedon"]==id,] <-
min(hz$hzdept[cond1])
        diags[["placic"]]["bottom_depth"][diags[["placic"]]["pedon"]==id,] <-
max(hz$hzdepb[cond1])
        diags[["placic"]]["thick"][diags[["placic"]]["pedon"]==id,] <-
max(hz$hzdepb[cond1]) - min(hz$hzdept[cond1])

      } else {

        diags[["placic"]]["cond3"][diags[["placic"]]["pedon"]==id,] <- FALSE

      }

    } else {

      diags[["placic"]]["cond3"][diags[["placic"]]["pedon"]==id,] <- TRUE
      diags[["placic"]]["hz"][diags[["placic"]]["pedon"]==id,] <- TRUE
      diags[["placic"]]["top_depth"][diags[["placic"]]["pedon"]==id,] <-
min(hz$hzdept[cond1])
      diags[["placic"]]["bottom_depth"][diags[["placic"]]["pedon"]==id,] <-
max(hz$hzdepb[cond1])
```

```
      diags[["placic"]]["thick"][diags[["placic"]]["pedon"]==id,] <-
max(hz$hzdepb[cond1]) - min(hz$hzdept[cond1])

    }


  } else {


    diags[["placic"]]["cond3"][diags[["placic"]]["pedon"]==id,] <- FALSE

  }


  return(diags)


}









################################################################################
# TODO:
# - add condition to check input EC units and convert as needed
# - check if sequential
salic <- function(hzs, diags, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)

  # EC must be in units of dS/m; check input and convert
  cond1 <- which(hz$ec >= 30)

  if(length(cond1)==0) {

    diags[["salic"]]["cond1"][diags[["salic"]]["pedon"]==id,] <- FALSE
    diags[["salic"]]["hz"][diags[["salic"]]["pedon"]==id,] <- FALSE
    return(diags)

  }


  diags[["salic"]]["cond1"][diags[["salic"]]["pedon"]==id,] <- TRUE


  thk <- sum(hz$hzthk[cond1])
  cond2 <- (hz$ec[cond1] * thk) >= 900
```

```
  if(cond2) {

    diags[["salic"]]["cond2"][diags[["salic"]]["pedon"]==id,] <- TRUE
    diags[["salic"]]["hz"][diags[["salic"]]["pedon"]==id,] <- TRUE

    if(thk >= 15) {

      diags[["salic"]]["cond0"][diags[["salic"]]["pedon"]==id,] <- TRUE
      diags[["salic"]]["hz"][diags[["salic"]]["pedon"]==id,] <- TRUE
      diags[["salic"]]["top_depth"][diags[["salic"]]["pedon"]==id,] <-
min(hz$hzdept[cond1])
      diags[["salic"]]["bottom_depth"][diags[["salic"]]["pedon"]==id,] <-
max(hz$hzdepb[cond1])
      diags[["salic"]]["thick"][diags[["salic"]]["pedon"]==id,] <-
max(hz$hzdepb[cond1]) - min(hz$hzdept[cond1])

    } else {

      diags[["salic"]]["cond0"][diags[["salic"]]["pedon"]==id,] <- FALSE
      diags[["salic"]]["hz"][diags[["salic"]]["pedon"]==id,] <- FALSE

    }


  } else {

    diags[["salic"]]["cond2"][diags[["salic"]]["pedon"]==id,] <- FALSE
    diags[["salic"]]["hz"][diags[["salic"]]["pedon"]==id,] <- FALSE

  }


  return(diags)


}
```

```
##############################################################################
# TODO:
# - add conditions for Sombric
# - skip for now
sombric <- function(hzs, diags, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
```

```
  return(diags)

}









################################################################################
# TODO:
# - check if beneath O, A, Ap, or E
spodic <- function(hzs, diags, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)


  # logical for if spodic mat even present
  spodic_mat <- as.logical(lapply(diags["spodic_mat"],function(x){x[["prop"]]
[which(x[["pedon"]]==id)]})) |
    any(hz$spodic_mat_pct > 0 )

  if(!spodic_mat) {

    diags[["spodic"]]["hz"][diags[["spodic"]]["pedon"]==id,] <- FALSE
    return(diags)

  }

  # spodic_mat_pct is the percent of the hz that is spodic mat
  # checker should set to 100 if none given but logical is true
  cond0 <- which(hz$spodic_mat_pct > 85 & hz$desgnmaster != "Ap")

  if(length(cond0)==0) {

    diags[["spodic"]]["hz"][diags[["spodic"]]["pedon"]==id,] <- FALSE
    diags[["spodic"]]["cond0"][diags[["spodic"]]["pedon"]==id,] <- FALSE

  }

  diags[["spodic"]]["cond0"][diags[["spodic"]]["pedon"]==id,] <- TRUE


  thk <- sum(hz$hzthk[cond0])

  if(thk > 2.5) {

    diags[["spodic"]]["hz"][diags[["spodic"]]["pedon"]==id,] <- TRUE
    diags[["spodic"]]["cond1"][diags[["spodic"]]["pedon"]==id,] <- TRUE
```

```
    diags[["spodic"]]["top_depth"][diags[["spodic"]]["pedon"]==id,] <-
min(hz$hzdept[cond0])
    diags[["spodic"]]["bottom_depth"][diags[["spodic"]]["pedon"]==id,] <-
max(hz$hzdepb[cond0])
    diags[["spodic"]]["thick"][diags[["spodic"]]["pedon"]==id,] <-
max(hz$hzdepb[cond0]) - min(hz$hzdept[cond0])

  } else {

    diags[["spodic"]]["hz"][diags[["spodic"]]["pedon"]==id,] <- FALSE
    diags[["spodic"]]["cond1"][diags[["spodic"]]["pedon"]==id,] <- FALSE

  }


  return(diags)

}









################################################################################
# TODO:
# - skip for now
sulfuric <- function(hzs, diags, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)

  return(diags)

}









################################################################################
# TODO:
# - check for Cd in checker and then set densic flag
densic <- function(hzs, diags, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
```

```
  cond <- which(hz$densic)

  if(length(cond)==0) {

    diags[["densic"]]["hz"][diags[["densic"]]["pedon"]==id,] <- FALSE
    return(diags)

  }

  diags[["densic"]]["hz"][diags[["densic"]]["pedon"]==id,] <- TRUE
  diags[["densic"]]["top_depth"][diags[["densic"]]["pedon"]==id,] <-
min(hz$hzdept[cond])
  diags[["densic"]]["bottom_depth"][diags[["densic"]]["pedon"]==id,] <-
max(hz$hzdepb[cond])
  diags[["densic"]]["thick"][diags[["densic"]]["pedon"]==id,] <-
max(hz$hzdepb[cond]) - min(hz$hzdept[cond])

  return(diags)


}









################################################################################
# TODO:
# - check for R in checker and then set lithic flag
lithic <- function(hzs, diags, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)

  cond <- which(hz$lithic)

  if(length(cond)==0) {

    diags[["lithic"]]["hz"][diags[["lithic"]]["pedon"]==id,] <- FALSE
    return(diags)

  }

  diags[["lithic"]]["hz"][diags[["lithic"]]["pedon"]==id,] <- TRUE
  diags[["lithic"]]["top_depth"][diags[["lithic"]]["pedon"]==id,] <-
min(hz$hzdept[cond])
  diags[["lithic"]]["bottom_depth"][diags[["lithic"]]["pedon"]==id,] <-
max(hz$hzdepb[cond])
```

```
  diags[["lithic"]]["thick"][diags[["lithic"]]["pedon"]==id,] <-
max(hz$hzdepb[cond]) - min(hz$hzdept[cond])

  return(diags)


}









################################################################################
# TODO:
# - check for Cr in checker and then set paralithic flag
paralithic <- function(hzs, diags, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)

  cond0 <- which(hz$paralithic)

  if(length(cond0)==0) {

    diags[["paralithic"]]["hz"][diags[["paralithic"]]["pedon"]==id,] <- FALSE
    return(diags)

  }

  diags[["paralithic"]]["hz"][diags[["paralithic"]]["pedon"]==id,] <- TRUE
  diags[["paralithic"]]["top_depth"][diags[["paralithic"]]["pedon"]==id,] <-
min(hz$hzdept[cond0])
  diags[["paralithic"]]["bottom_depth"][diags[["paralithic"]]["pedon"]==id,]
<- max(hz$hzdepb[cond0])
  diags[["paralithic"]]["thick"][diags[["paralithic"]]["pedon"]==id,] <-
max(hz$hzdepb[cond0]) - min(hz$hzdept[cond0])

  return(diags)


}
```

```
#############################################################################
# TODO:
# - assumes 0% OM
# - revise to check for sm
petroferric <- function(hzs, diags, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)

  cond <- which(hz$cemented & hz$cemagt == "iron" & hz$om == 0)


  if(length(cond)==0) {

    diags[["petroferric"]]["hz"][diags[["petroferric"]]["pedon"]==id,] <-
FALSE
    return(diags)

  }

  diags[["petroferric"]]["hz"][diags[["petroferric"]]["pedon"]==id,] <- TRUE
  diags[["petroferric"]]["top_depth"][diags[["petroferric"]]["pedon"]==id,] <-
min(hz$hzdept[cond])
  diags[["petroferric"]]["bottom_depth"][diags[["petroferric"]]["pedon"]==id,]
<- max(hz$hzdepb[cond])
  diags[["petroferric"]]["thick"][diags[["petroferric"]]["pedon"]==id,] <-
max(hz$hzdepb[cond]) - min(hz$hzdept[cond])

  return(diags)


}









#############################################################################
# MUST NOT HAVE MORE THAN ONE OF THE DIAGNOSTIC HZ LISTED
# TODO:
# - consider more than one abrupt textural change
abrupt_text <- function(hzs, diags, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)



  ## check if diags hz are even present
```

```
  diag_hz <- c("argillic","glossic","kandic","natric")

  cond1 <- any(as.logical(lapply(diags[diag_hz],function(x){x[["hz"]]
[which(x[["pedon"]]==id)]})))
  idx <- as.logical(lapply(diags[diag_hz],function(x){x[["hz"]]
[which(x[["pedon"]]==id)]}))
  l <- lapply(diags[diag_hz],function(x){x[["hz"]][which(x[["pedon"]]==id)]})

  if(!cond1) {

    diags[["abrupt_text"]]["prop"][diags[["abrupt_text"]]["pedon"]==id,] <-
FALSE
    diags[["abrupt_text"]]["cond1"][diags[["abrupt_text"]]["pedon"]==id,] <-
FALSE
    return(diags)

  }




  cond1_diag <- names(l)[which(idx)]


  ## check if bottom of diags hz is at bottom of profile
  # if(max(hz$hzdepb) == diags[[cond1_diag]]["bottom_depth"]
[diags[[cond1_diag]]["pedon"]==id,]) {
  #
  #   diags[["abrupt_text"]]["prop"][diags[["abrupt_text"]]["pedon"]==id,] <-
FALSE
  #   diags[["abrupt_text"]]["cond2"][diags[["abrupt_text"]]["pedon"]==id,] <-
FALSE
  #   return(diags)
  #
  # }

  diag_bot <- diags[[cond1_diag]]["bottom_depth"][diags[[cond1_diag]]
["pedon"]==id,]
  diag_top <- diags[[cond1_diag]]["top_depth"][diags[[cond1_diag]]
["pedon"]==id,]


  cond1_hz <- prof_slice(bottom = diag_bot,
                         top = diag_top,
                         hz = hz)


  cond1 <- cond1_hz$claytotal >= 8

  if(!all(cond1)) {

    diags[["abrupt_text"]]["prop"][diags[["abrupt_text"]]["pedon"]==id,] <-
FALSE
```

```
    diags[["abrupt_text"]]["cond1"][diags[["abrupt_text"]]["pedon"]==id,] <-
FALSE
    return(diags)

  } else {

    diags[["abrupt_text"]]["cond1"][diags[["abrupt_text"]]["pedon"]==id,] <-
TRUE

  }

  # get clay content of first horizon of the diags hz
  diag_clay <- cond1_hz$claytotal[1]

  diag_slice <- cond1_hz


  epi_top <- diags[["epipedon"]]["top_depth"][diags[["epipedon"]]
["pedon"]==id,]
  epi_bot <- diags[["epipedon"]]["bottom_depth"][diags[["epipedon"]]
["pedon"]==id,]

  epi_slice <- prof_slice(bottom = epi_bot,
                          top = epi_top,
                          hz = hz)


  eluv_top <- min(hz$hzdept)
  eluv_bot <- diag_top

  eluv_slice <- prof_slice(bottom = eluv_bot,
                           top = eluv_top,
                           hz = hz)

  # get clay of last hz of epipedon
  epi_clay <- epi_slice$claytotal[nrow(epi_slice)]

  # get clay of hz directly above diags hz
  eluv_clay <- eluv_slice$claytotal[nrow(eluv_slice)]

  # check clay increase from bottom of epipedon
  if(epi_clay < 20) {

    # this should find the first horizon within 7.5cm of the bottom of the
eluv
    # this could give multiple horizons
    vd_slice <- prof_slice(bottom = epi_bot+7.5,
                           top = epi_bot,
                           hz=hz)

    vd_clay <- vd_slice$claytotal

    if(any(vd_clay >= epi_clay*2)) {
```

```
      vd_idx <- which(vd_clay >= epi_clay*2)

      diags[["abrupt_text"]]["prop"][diags[["abrupt_text"]]["pedon"]==id,] <-
TRUE
      diags[["abrupt_text"]]["cond2a"][diags[["abrupt_text"]]["pedon"]==id,]
<- TRUE
      diags[["abrupt_text"]]["top_depth"][diags[["abrupt_text"]]
["pedon"]==id,] <- epi_bot
      diags[["abrupt_text"]]["bottom_depth"][diags[["abrupt_text"]]
["pedon"]==id,] <- max(vd_slice$hzdepb[vd_idx])
      diags[["abrupt_text"]]["thick"][diags[["abrupt_text"]]["pedon"]==id,] <-
max(vd_slice$hzdepb[vd_idx]) - epi_bot
      return(diags)

    }

  } else {

    vd_slice <- prof_slice(bottom = epi_bot+7.5,
                           top = epi_bot,
                           hz=hz)

    vd_clay <- vd_slice$claytotal

    if(any(vd_clay >= epi_clay+20) & any(diag_slice$claytotal >= epi_clay*2))
{

      vd_idx <- which(vd_clay >= epi_clay+20)

      diags[["abrupt_text"]]["prop"][diags[["abrupt_text"]]["pedon"]==id,] <-
TRUE
      diags[["abrupt_text"]]["cond2b"][diags[["abrupt_text"]]["pedon"]==id,]
<- TRUE
      diags[["abrupt_text"]]["top_depth"][diags[["abrupt_text"]]
["pedon"]==id,] <- epi_bot
      # diags[["abrupt_text"]]["bottom_depth"][diags[["abrupt_text"]]
["pedon"]==id,] <- max(vd_slice$hzdepb[vd_idx])
      # diags[["abrupt_text"]]["thick"][diags[["abrupt_text"]]["pedon"]==id,]
<- max(vd_slice$hzdepb[vd_idx]) - epi_bot
      return(diags)


    }

  }




  if(eluv_clay < 20) {

      # check clay increase from bottom of the eluvial horizon
```

```
      # this should find the first horizon within 7.5cm of the bottom of the
eluv
      # this could give multiple horizons
      vd_slice <- prof_slice(bottom = eluv_bot+7.5,
                             top = eluv_bot,
                             hz=hz)

      vd_clay <- vd_slice$claytotal

      if(any(vd_clay >= eluv_clay*2)) {

        vd_idx <- which(vd_clay >= eluv_clay*2)

        diags[["abrupt_text"]]["prop"][diags[["abrupt_text"]]["pedon"]==id,]
<- TRUE
        diags[["abrupt_text"]]["cond2a"][diags[["abrupt_text"]]["pedon"]==id,]
<- TRUE
        diags[["abrupt_text"]]["top_depth"][diags[["abrupt_text"]]
["pedon"]==id,] <- eluv_bot
        # diags[["abrupt_text"]]["bottom_depth"][diags[["abrupt_text"]]
["pedon"]==id,] <- max(vd_slice$hzdepb[vd_idx])
        # diags[["abrupt_text"]]["thick"][diags[["abrupt_text"]]
["pedon"]==id,] <- max(vd_slice$hzdepb[vd_idx]) - eluv_bot
        return(diags)

      }


  } else {


      if(any(vd_clay >= eluv_clay+20) & any(diag_slice$claytotal >=
eluv_clay*2)) {

      vd_idx <- which(vd_clay >= eluv_clay+20)

      diags[["abrupt_text"]]["prop"][diags[["abrupt_text"]]["pedon"]==id,] <-
TRUE
      diags[["abrupt_text"]]["cond2b"][diags[["abrupt_text"]]["pedon"]==id,]
<- TRUE
      diags[["abrupt_text"]]["top_depth"][diags[["abrupt_text"]]
["pedon"]==id,] <- eluv_bot
      # diags[["abrupt_text"]]["bottom_depth"][diags[["abrupt_text"]]
["pedon"]==id,] <- max(vd_slice$hzdepb[vd_idx])
      # diags[["abrupt_text"]]["thick"][diags[["abrupt_text"]]["pedon"]==id,]
<- max(vd_slice$hzdepb[vd_idx]) - eluv_bot
      return(diags)

      }
  }


  diags[["abrupt_text"]]["prop"][diags[["abrupt_text"]]["pedon"]==id,] <-
FALSE
```

```
  diags[["abrupt_text"]]["cond2"][diags[["abrupt_text"]]["pedon"]==id,] <-
FALSE
  diags[["abrupt_text"]]["cond2a"][diags[["abrupt_text"]]["pedon"]==id,] <-
FALSE
  diags[["abrupt_text"]]["cond2b"][diags[["abrupt_text"]]["pedon"]==id,] <-
FALSE
  return(diags)



}









################################################################################
# TODO:
# - figure out phosphorus vs phosphate retention
# - figure out proxy for cond3a; use siltco + sandtotal?
# - update hzs andic flag
andic_properties <- function(hzs, diags, id) {

  hzs_idx <- which(hzs$pedon_id == id)
  hz <- dplyr::filter(hzs, pedon_id==id)


  ## Condition 1: organic carbon
  cond1 <- which(hz$oc < 25)


  ## Cond 2: a,b,c

  ## cond 2a
  cond2a <- cond1[cond1 %in% which(hz$dbthirdbar <= 0.9)]

  if(length(cond2a)==0) {

    diags[["andic_prop"]]["cond2a"][diags[["andic_prop"]]["pedon"]==id,] <-
FALSE

  }


  ## cond 2b
  cond2b <- cond2a[cond2a %in% which(hz$p_ret >= 85)]

  if(length(cond2b)==0) {
```

```
    diags[["andic_prop"]]["cond2b"][diags[["andic_prop"]]["pedon"]==id,] <-
FALSE

  }


  ## cond 2c
  cond2c <- cond2b[cond2b %in% which((hz$aloxalate + 0.5*hz$feoxalate) >= 2)]

  if(length(cond2c)==0) {

    diags[["andic_prop"]]["cond2c"][diags[["andic_prop"]]["pedon"]==id,] <-
FALSE

  }


  ## cond 2 overall
  if(length(cond2a)==0 | length(cond2b)==0 | length(cond2c)==0) {

    cond2 <- FALSE
    andic_hz_2 <- vector()
    diags[["andic_prop"]]["cond2"][diags[["andic_prop"]]["pedon"]==id,] <-
FALSE

  } else {

    cond2 <- TRUE
    andic_hz_2 <- cond2c
    diags[["andic_prop"]]["cond2"][diags[["andic_prop"]]["pedon"]==id,] <-
TRUE

  }




  ## cond 3: a,b,c,d,e

  ## cond 3a
  cond3a <- which((hz$sandtotal+hz$siltco) > 30)

  if(length(cond3a)==0) {

    diags[["andic_prop"]]["cond3a"][diags[["andic_prop"]]["pedon"]==id,] <-
FALSE

  }


  ## cond 3b
  cond3b <- cond3a[cond3a %in% which(hz$p_ret >= 25)]

  if(length(cond3b)==0) {
```

```
    diags[["andic_prop"]]["cond3b"][diags[["andic_prop"]]["pedon"]==id,] <-
FALSE

  }


  ## cond 3c
  cond3c <- cond3b[cond3b %in% which((hz$aloxalate + 0.5*hz$feoxalate) >=
0.4)]

  if(length(cond3c)==0) {

    diags[["andic_prop"]]["cond3c"][diags[["andic_prop"]]["pedon"]==id,] <-
FALSE

  }


  ## cond 3d
  cond3d <- cond3c[cond3c %in% which(hz$vglass >= 5)]

  if(length(cond3d)==0) {

    diags[["andic_prop"]]["cond3d"][diags[["andic_prop"]]["pedon"]==id,] <-
FALSE

  }


  ## cond 3e
  cond3e <- cond3d[cond3d %in% which((((hz$aloxalate + 0.5*hz$feoxalate) * 15.
625) + hz$vglass) >= 36.25)]

  if(length(cond3e)==0) {

    diags[["andic_prop"]]["cond3e"][diags[["andic_prop"]]["pedon"]==id,] <-
FALSE

  }



  ## cond 3 overall
  if(length(cond3a)==0 | length(cond3b)==0 | length(cond3c)==0 |
length(cond3d)==0 | length(cond3e)==0) {

    cond3 <- FALSE
    diags[["andic_prop"]]["cond3"][diags[["andic_prop"]]["pedon"]==id,] <-
FALSE

  } else {

    andic_hz_3 <- cond3e
```

```
    cond3 <- TRUE
    diags[["andic_prop"]]["cond3"][diags[["andic_prop"]]["pedon"]==id,] <-
TRUE

  }



  if(cond2 | cond3) {

    hzs_andic <- hzs_idx[unique(c(andic_hz_2,andic_hz_3))]
    hzs$andic[hzs_andic] <- TRUE
    diags[["andic_prop"]]["prop"][diags[["andic_prop"]]["pedon"]==id,] <- TRUE

  } else {

    diags[["andic_prop"]]["prop"][diags[["andic_prop"]]["pedon"]==id,] <-
FALSE

  }


  return(list(hzs=hzs,diags=diags))

}









################################################################################
# TODO:
# - cond1/2/3 chroma wet or dry? checking either
# - need location of albic material? which one preferred?
albic_mat <- function(hzs, diags, id) {

  hzs_idx <- which(hzs$pedon_id == id)
  hz <- dplyr::filter(hzs, pedon_id==id)



  cond1_log <- FALSE
  cond2_log <- FALSE
  cond3_log <- FALSE


  cond1 <- which(hz$colorchroma_d <= 2 | hz$colorchroma_w <= 2)

  if(length(cond1)==0) {
```

```
    diags[["albic_mat"]]["cond1"][diags[["albic_mat"]]["pedon"]==id,] <- FALSE

  }


  cond1a <- cond1[cond1 %in% which(hz$colorvalue_w == 3 & hz$colorvalue_d >=
6)]

  cond1b <- cond1[cond1 %in% which(hz$colorvalue_w >= 4 & hz$colorvalue_d >=
5)]


  if(length(cond1a)==0) {

    diags[["albic_mat"]]["cond1a"][diags[["albic_mat"]]["pedon"]==id,] <-
FALSE

  } else {

    cond1 <- cond1a
    cond1_log <- TRUE
    diags[["albic_mat"]]["cond1"][diags[["albic_mat"]]["pedon"]==id,] <- TRUE
    diags[["albic_mat"]]["cond1a"][diags[["albic_mat"]]["pedon"]==id,] <- TRUE

  }


  if(length(cond1b)==0) {

    diags[["albic_mat"]]["cond1b"][diags[["albic_mat"]]["pedon"]==id,] <-
FALSE

  } else {

    cond1 <- cond1b
    cond1_log <- TRUE
    diags[["albic_mat"]]["cond1"][diags[["albic_mat"]]["pedon"]==id,] <- TRUE
    diags[["albic_mat"]]["cond1a"][diags[["albic_mat"]]["pedon"]==id,] <- TRUE

  }




  cond2 <- which(hz$colorchroma_d <= 3 | hz$colorchroma_w <= 3)

  if(length(cond2)==0) {

    diags[["albic_mat"]]["cond2"][diags[["albic_mat"]]["pedon"]==id,] <- FALSE

  }
```

```
  cond2a <- cond2[cond2 %in% which(hz$colorvalue_w >= 6)]

  cond2b <- cond2[cond2 %in% which(hz$colorvalue_d >= 7)]


  if(length(cond2a)==0) {

    diags[["albic_mat"]]["cond2a"][diags[["albic_mat"]]["pedon"]==id,] <-
FALSE

  } else {

    cond2 <- cond2a
    cond2_log <- TRUE
    diags[["albic_mat"]]["cond2"][diags[["albic_mat"]]["pedon"]==id,] <- TRUE
    diags[["albic_mat"]]["cond2a"][diags[["albic_mat"]]["pedon"]==id,] <- TRUE

  }


  if(length(cond2b)==0) {

    diags[["albic_mat"]]["cond2b"][diags[["albic_mat"]]["pedon"]==id,] <-
FALSE

  } else {

    cond2 <- cond2b
    cond2_log <- TRUE
    diags[["albic_mat"]]["cond2"][diags[["albic_mat"]]["pedon"]==id,] <- TRUE
    diags[["albic_mat"]]["cond2a"][diags[["albic_mat"]]["pedon"]==id,] <- TRUE

  }



  cond3_hue <- c("5YR","2.5YR","10R","7.5R","5R","2.5R")
  cond3 <- which(hz$colorhue_w %in% cond3_hue)


  if(length(cond3)==0) {

    diags[["albic_mat"]]["cond3"][diags[["albic_mat"]]["pedon"]==id,] <- FALSE

  }



  cond3a <- cond3[cond3 %in% which(hz$colorvalue_w == 3 & hz$colorvalue_d >=
6)]

  cond3b <- cond3[cond3 %in% which(hz$colorvalue_w >= 4 & hz$colorvalue_d >=
5)]
```

```
  if(length(cond3a)==0) {

    diags[["albic_mat"]]["cond3a"][diags[["albic_mat"]]["pedon"]==id,] <-
FALSE

  } else {

    cond3 <- cond3a
    cond3_log <- TRUE
    diags[["albic_mat"]]["cond3"][diags[["albic_mat"]]["pedon"]==id,] <- TRUE
    diags[["albic_mat"]]["cond3a"][diags[["albic_mat"]]["pedon"]==id,] <- TRUE

  }


  if(length(cond3b)==0) {

    diags[["albic_mat"]]["cond3b"][diags[["albic_mat"]]["pedon"]==id,] <-
FALSE

  } else {

    cond3 <- cond3b
    cond3_log <- TRUE
    diags[["albic_mat"]]["cond3"][diags[["albic_mat"]]["pedon"]==id,] <- TRUE
    diags[["albic_mat"]]["cond3a"][diags[["albic_mat"]]["pedon"]==id,] <- TRUE

  }



  if(any(c(cond1_log, cond2_log, cond3_log))) {

    l <- list(cond1,cond2,cond3)
    idx <- unlist(l[which(c(cond1_log, cond2_log, cond3_log))])
    top_depth <- min(hz$hzdept[idx])
    bottom_depth <- max(hz$hzdepb[idx])

    hzs$albic_vol[hzs_idx[idx]] <- 100

    diags[["albic_mat"]]["prop"][diags[["albic_mat"]]["pedon"]==id,] <- TRUE
    diags[["albic_mat"]]["top_depth"][diags[["albic_mat"]]["pedon"]==id,] <-
top_depth
    diags[["albic_mat"]]["bottom_depth"][diags[["albic_mat"]]["pedon"]==id,]
<- bottom_depth
    diags[["albic_mat"]]["thick"][diags[["albic_mat"]]["pedon"]==id,] <-
bottom_depth - top_depth


  } else {

    diags[["albic_mat"]]["prop"][diags[["albic_mat"]]["pedon"]==id,] <- FALSE
```

```
  }

  return(list(hzs=hzs,diags=diags))


}
```

```
#############################################################################
# TODO:
# - check for temperature conditions?
# - SKIP FOR NOW
anhydrous_cond <- function(hzs, diags, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)

  return(diags)

}
```

```
#############################################################################
# TODO:
# - SKIP FOR NOW
fragic_prop <- function(hzs, diags, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)

  return(diags)

}
```

```
#############################################################################
# TODO:
# - revise cemagt
# - cond2b2, cond2b4 not checked
# - check is spodic_mat in epipedon or other feature?
spodic_mat <- function(hzs, diags, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)


  ## cond 1
  cond1 <- which(hz$ph1to1h2o <= 5.9 & hz$oc >= 0.6)

  if(length(cond1)==0) {

    diags[["spodic_mat"]]["cond1"][diags[["spodic_mat"]]["pedon"]==id,] <-
FALSE
    diags[["spodic_mat"]]["prop"][diags[["spodic_mat"]]["pedon"]==id,] <-
FALSE
    return(diags)

  }




  ## cond 2a: is there an albic horizon above with certain colors

  albic_log <- diags[["albic"]]["hz"][diags[["albic"]]["pedon"]==id,]

  if(!albic_log) {

    cond2a <- FALSE
    diags[["spodic_mat"]]["cond2a"][diags[["spodic_mat"]]["pedon"]==id,] <-
FALSE

  } else {

    # logical
    albic_abv <- diags[["albic"]]["bottom_depth"][diags[["albic"]]
["pedon"]==id,] > min(hz$hzdept[cond1])

    if(!albic_abv) {

      cond2a <- FALSE
      diags[["spodic_mat"]]["cond2a"][diags[["spodic_mat"]]["pedon"]==id,] <-
FALSE

    } else {
```

```
      cond2a <- TRUE

    }


  }

  ## check criteria for the rest of 2a if an albic horizon is present and
above cond1
  if(cond2a) {

    albic_bot <- diags[["albic"]]["bottom_depth"][diags[["albic"]]
["pedon"]==id,]

    ## hz idx below albic, should be length 1
    hz_bl <- max(which(hz$hzdept < albic_bot+1))


    cond2a1 <- (hz$colorhue_w[hz_bl] %in%
c("5YR","2.5YR","10R","7.5R","5R","2.5R"))

    if(is.na(cond2a1) | !cond2a1) {

      diags[["spodic_mat"]]["cond2a1"][diags[["spodic_mat"]]["pedon"]==id,] <-
FALSE

    } else {

      diags[["spodic_mat"]]["cond2a1"][diags[["spodic_mat"]]["pedon"]==id,] <-
TRUE

    }



    cond2a2 <- (hz$colorhue_w[hz_bl] == "7.5YR" & hz$colorvalue_w[hz_bl] <= 5
& hz$colorchroma_w[hz_bl] <= 4)

    if(is.na(cond2a2) | !cond2a2) {

      diags[["spodic_mat"]]["cond2a2"][diags[["spodic_mat"]]["pedon"]==id,] <-
FALSE

    } else {

      diags[["spodic_mat"]]["cond2a2"][diags[["spodic_mat"]]["pedon"]==id,] <-
TRUE

    }


    cond2a3 <- (hz$colorhue_w[hz_bl] %in% c("10YR","N") &
hz$colorvalue_w[hz_bl] <= 2 & hz$colorchroma_w[hz_bl] <= 2)
```

```
    if(is.na(cond2a3) | !cond2a3) {

      diags[["spodic_mat"]]["cond2a3"][diags[["spodic_mat"]]["pedon"]==id,] <-
FALSE

    } else {

      diags[["spodic_mat"]]["cond2a3"][diags[["spodic_mat"]]["pedon"]==id,] <-
TRUE

    }



    cond2a4 <- (hz$color_d[hz_bl] == "10YR 3/1")

    if(is.na(cond2a4) | !cond2a4) {

      diags[["spodic_mat"]]["cond2a4"][diags[["spodic_mat"]]["pedon"]==id,] <-
FALSE

    } else {

      diags[["spodic_mat"]]["cond2a4"][diags[["spodic_mat"]]["pedon"]==id,] <-
TRUE

    }



    if(any(c(cond2a1,cond2a2,cond2a3,cond2a4),na.rm=TRUE)) {

      cond2a <- TRUE
      diags[["spodic_mat"]]["cond2a"][diags[["spodic_mat"]]["pedon"]==id,] <-
TRUE

    } else {

      cond2a <- FALSE
      diags[["spodic_mat"]]["cond2a"][diags[["spodic_mat"]]["pedon"]==id,] <-
FALSE

    }


  }




  ## cond2b
```

```
  cond2b <- cond1[cond1 %in% which((hz$colorhue_w %in%
c("5YR","2.5YR","10R","7.5R","5R","2.5R")) |
                                   (hz$colorhue_w == "7.5YR" & hz$colorvalue_w
<= 5 & hz$colorchroma_w <= 4) |
                                   (hz$colorhue_w %in% c("10YR","N") &
hz$colorvalue_w <= 2 & hz$colorchroma_w <= 2) |
                                   (hz$color_d == "10YR 3/1") |
                                   (hz$colorhue_w == "7.5YR" & hz$colorvalue_w
<= 5 & hz$colorchroma_w %in% c(5,6)))]

  if(length(cond2b)==0) {

    cond2b <- FALSE
    diags[["spodic_mat"]]["cond2b"][diags[["spodic_mat"]]["pedon"]==id,] <-
FALSE

  } else {

      cond2b1 <- cond2b[cond2b %in% which(hz$cemented & hz$cemagt %in%
c("organic matter","al") &
                                          hz$rrc %in% c("very firm","extremely
firm","slightly rigid","rigid","very rigid"))]

      if(length(cond2b1)==0) {

        diags[["spodic_mat"]]["cond2b1"][diags[["spodic_mat"]]["pedon"]==id,]
<- FALSE

      } else {

        cond2b1 <- TRUE

      }


      ## cond2b2 SKIPPED FOR NOW


      diag_hz <- c("umbric","ochric","albic")

      cond2b3_diag <- which(as.logical(lapply(diags[diag_hz],function(x)
{x[["hz"]][which(x[["pedon"]]==id)]})))

      if(length(cond2b3_diag)==0) {

        cond2b3 <- vector()
        diags[["spodic_mat"]]["cond2b3"][diags[["spodic_mat"]]["pedon"]==id,]
<- FALSE

      } else {

        ## just take the first one
        cond2b3_diag <- cond2b3_diag[1]
```

```
        epi_top <- dplyr::filter(diags[[diag_hz[cond2b3_diag]]],pedon==id)
$top_depth
        epi_bot <- dplyr::filter(diags[[diag_hz[cond2b3_diag]]],pedon==id)
$bottom_depth
        epi_hz <- prof_slice(epi_bot,epi_top,hz)
        epi_alfe <- epi_hz$aloxalate + 0.5*epi_hz$feoxalate

        hz_alfe <- (hz$aloxalate + 0.5*hz$feoxalate)
        # cond2b3 <- cond2b[cond2b %in% which(hz_alfe >= 0.5 & epi_alfe <=
0.5*hz_alfe)]
        hz_alfe_wa <- weighted.mean(hz_alfe[cond2b],hz$hzthk[cond2b]/
sum(hz$hzthk[cond2b]))
        cond2b3 <- all(hz_alfe[cond2b] >=0.5) & all(epi_alfe <=
0.5*hz_alfe_wa)

        if(!cond2b3) {

          diags[["spodic_mat"]]["cond2b3"][diags[["spodic_mat"]]
["pedon"]==id,] <- FALSE

        }

      }


      ## cond2b4 SKIPPED FOR NOW


      #lst <- list(cond2b1,cond2b3)

      #if(any(lengths(lst)!=0)) {
      if(any(cond2b1,cond2b3)) {

        cond2b <- TRUE
        diags[["spodic_mat"]]["cond2b"][diags[["spodic_mat"]]["pedon"]==id,]
<- TRUE

      }

  }


  if(any(c(cond2a,cond2b))) {

    diags[["spodic_mat"]]["prop"][diags[["spodic_mat"]]["pedon"]==id,] <- TRUE

  } else {

    diags[["spodic_mat"]]["prop"][diags[["spodic_mat"]]["pedon"]==id,] <-
FALSE

  }
```

```
  return(diags)


}









################################################################################
# TODO:
# This determines the bounds for the organic and mineral parts of the pedon.
# - What to do with buried organic horizons
orgmin <- function(hzs, diags, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)


  ## organic bottom
  orghz <- which(hz$desgnmaster == "O")

  if(!is.sequential(orghz)) {

    top_seq <- get_seq(orghz)

  } else {

    top_seq <- orghz

  }

  if(length(top_seq)!=0) {

    org_top <- min(hz$hzdept[top_seq])
    org_bot <- max(hz$hzdepb[top_seq])
    diags[["orgmin"]]["org_top"][diags[["orgmin"]]["pedon"]==id,] <- org_top
    diags[["orgmin"]]["org_bottom"][diags[["orgmin"]]["pedon"]==id,] <-
org_bot

  }









  ## mineral surface
  minhz <- which(hz$desgnmaster != "O")
```

```
  if(length(minhz)!=0) {

    first_min <- min(minhz)
    min_top <- hz$hzdept[first_min]

  } else {

    first_min <- max(hz$hzdepb) # assume mineral starts under organic
    min_top <- first_min

  }



  diags[["orgmin"]]["min_top"][diags[["orgmin"]]["pedon"]==id,] <- min_top


  return(diags)


}









################################################################################
# TODO:
# - changed bulk density to oven dry
# This determines the control section for Histosols and Histels
org_thk <- function(hzs, diags, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)


  ## Surface Tier ##

  ## if st1 OR st2, surface tier is 60cm; otherwise 30cm
  st1_sl <- prof_slice(60,hz$hzdept[1],hz)

  st1 <- all(grepl('Oi',st1_sl$hzname) & st1_sl$sphgm_vol > 75)
  st2 <- all(st1_sl$dbovendry <= 0.1)

  if(is.na(st2)) {

    st2 <- FALSE

  }
```

```
  org_top <- 0
  diags[["org_thk"]]["surface_tier_top"][diags[["org_thk"]]["pedon"]==id,] <-
0
  diags[["org_thk"]]["org_ctrl_top"][diags[["org_thk"]]["pedon"]==id,] <- 0

  if(st1 | st2) {

    st1_bot <- 60
    diags[["org_thk"]]["surface_tier_bottom"][diags[["org_thk"]]
["pedon"]==id,] <- 60


  } else {

    st1_bot <- 30
    diags[["org_thk"]]["surface_tier_bottom"][diags[["org_thk"]]
["pedon"]==id,] <- 30

  }


  if(length(which(hz$hzname=="W"))>0) {

    water <- hz$hzdept[which(hz$hzname=="W")]

  } else {

    water <- max(hz$hzdepb) + 1

  }



  ## if contacts shallower, the control section is truncated ##

  contacts <-
min(c(diag_top_depth(c("densic","lithic","paralithic"),diags,id),water),na.rm=TRUE)



  if(contacts <= st1_bot + 100) {

    subt_top <- st1_bot
    subt_bot <- contacts

    diags[["org_thk"]]["subsurface_tier_top"][diags[["org_thk"]]
["pedon"]==id,] <- subt_top
    diags[["org_thk"]]["subsurface_tier_bottom"][diags[["org_thk"]]
["pedon"]==id,] <- subt_bot

    diags[["org_thk"]]["org_ctrl_bottom"][diags[["org_thk"]]["pedon"]==id,] <-
subt_bot
```

```
  } else {

    subt_top <- st1_bot
    subt_bot <- st1_bot + 60
    btmt_top <- subt_bot
    btmt_bot <- subt_bot + 40

    diags[["org_thk"]]["subsurface_tier_top"][diags[["org_thk"]]
["pedon"]==id,] <- subt_top
    diags[["org_thk"]]["subsurface_tier_bottom"][diags[["org_thk"]]
["pedon"]==id,] <- subt_bot
    diags[["org_thk"]]["bottom_tier_top"][diags[["org_thk"]]["pedon"]==id,] <-
btmt_top
    diags[["org_thk"]]["bottom_tier_bottom"][diags[["org_thk"]]["pedon"]==id,]
<- btmt_bot

    diags[["org_thk"]]["org_ctrl_bottom"][diags[["org_thk"]]["pedon"]==id,] <-
btmt_bot

  }




  return(diags)


}









################################################################################
# TODO:
# This function determines the bounds of the particle-size control section in
mineral soils.
# This must be run AFTER the orders.
partctrl <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)


  ## check if shallower soil
  if(max(hz$hzdepb) < 36) {
```

```
    partctrl_top <- 0
    partctrl_bot <- max(hz$hzdepb)

    diags[["partctrl"]]["pc_top_depth"][diags[["partctrl"]]["pedon"]==id,] <-
partctrl_top
    diags[["partctrl"]]["pc_bottom_depth"][diags[["partctrl"]]["pedon"]==id,]
<- partctrl_bot


    return(diags)

  }






  ## check if andisol
  if(tax[["andisols"]]["tax"][tax[["andisols"]]["pedon"]==id,]) {

    partctrl_top <- 0

    if(any(hz$rll)) {

      partctrl_bot <- hz$hzdept[min(which(hz$rll))]

    } else if(max(hz$hzdepb < 100)){

      partctrl_bot <- max(hz$hzdepb)

    } else {

      partctrl_bot <- 100

    }


    diags[["partctrl"]]["pc_top_depth"][diags[["partctrl"]]["pedon"]==id,] <-
partctrl_top
    diags[["partctrl"]]["pc_bottom_depth"][diags[["partctrl"]]["pedon"]==id,]
<- partctrl_bot


    return(diags)

  }




  ## check for argillic, kandic, natric hz
```

```
  if(diags[["argillic"]]["hz"][diags[["argillic"]]["pedon"]==id,]) {

    partctrl_top <- diags[["argillic"]]["top_depth"][diags[["argillic"]]
["pedon"]==id,]

    if(diags[["argillic"]]["thick"][diags[["argillic"]]["pedon"]==id,] < 50) {

      partctrl_bot <- diags[["argillic"]]["bottom_depth"][diags[["argillic"]]
["pedon"]==id,]

    } else {

      partctrl_bot <- diags[["argillic"]]["top_depth"][diags[["argillic"]]
["pedon"]==id,] + 50

    }


    diags[["partctrl"]]["pc_top_depth"][diags[["partctrl"]]["pedon"]==id,] <-
partctrl_top
    diags[["partctrl"]]["pc_bottom_depth"][diags[["partctrl"]]["pedon"]==id,]
<- partctrl_bot


    return(diags)

  }

  if(diags[["kandic"]]["hz"][diags[["kandic"]]["pedon"]==id,]) {

    partctrl_top <- diags[["kandic"]]["top_depth"][diags[["kandic"]]
["pedon"]==id,]

    if(diags[["argillic"]]["thick"][diags[["argillic"]]["pedon"]==id,] < 50) {

      partctrl_bot <- diags[["kandic"]]["bottom_depth"][diags[["kandic"]]
["pedon"]==id,]

    } else {

      partctrl_bot <- diags[["kandic"]]["top_depth"][diags[["kandic"]]
["pedon"]==id,] + 50

    }


    diags[["partctrl"]]["pc_top_depth"][diags[["partctrl"]]["pedon"]==id,] <-
partctrl_top
    diags[["partctrl"]]["pc_bottom_depth"][diags[["partctrl"]]["pedon"]==id,]
<- partctrl_bot


    return(diags)
```

```
  }


  if(diags[["natric"]]["hz"][diags[["natric"]]["pedon"]==id,]) {

    partctrl_top <- diags[["natric"]]["top_depth"][diags[["natric"]]
["pedon"]==id,]

    if(diags[["natric"]]["thick"][diags[["natric"]]["pedon"]==id,] < 50) {

      partctrl_bot <- diags[["natric"]]["bottom_depth"][diags[["natric"]]
["pedon"]==id,]

    } else {

      partctrl_bot <- diags[["natric"]]["top_depth"][diags[["natric"]]
["pedon"]==id,] + 50

    }


    diags[["partctrl"]]["pc_top_depth"][diags[["partctrl"]]["pedon"]==id,] <-
partctrl_top
    diags[["partctrl"]]["pc_bottom_depth"][diags[["partctrl"]]["pedon"]==id,]
<- partctrl_bot


    return(diags)


  }




  ## with Ap and other

  if(any(hz$hzname == "Ap")) {

    deepest_ap <- min(which(hz$hzname == "Ap"))
    partctrl_top <- hz$hzdept[deepest_ap]
    partctrl_bot <- 100

  } else {

    partctrl_top <- 0
    partctrl_bot <- 100

  }
```

```
  diags[["partctrl"]]["pc_top_depth"][diags[["partctrl"]]["pedon"]==id,] <-
partctrl_top
  diags[["partctrl"]]["pc_bottom_depth"][diags[["partctrl"]]["pedon"]==id,] <-
partctrl_bot

  return(diags)



}
```

Appendix I:

orders.R

```
# This script contains the functions for classifying taxonomy at the Order
level.

# TODO:
# - for taxonomy functions, diag needs to be diags and filtered like hzs



################################################################################
gelisols <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  ## cond 1
  cond1 <- any(hz$permafrost == TRUE & hz$hzdept <= 100)

  if(cond1) {

    tax[["gelisols"]]["cond1"][tax[["gelisols"]]["pedon"]==id,] <- TRUE
    tax[["gelisols"]]["tax"][tax[["gelisols"]]["pedon"]==id,] <- TRUE

    return(tax)

  } else {

    tax[["gelisols"]]["cond1"][tax[["gelisols"]]["pedon"]==id,] <- FALSE

  }

  cond2_1 <- any(hz$gelic_mat == TRUE & hz$hzdept <= 100)
  cond2_2 <- any(hz$permafrost == TRUE & hz$hzdept <= 200)

  if(cond2_1 & cond2_2) {

    tax[["gelisols"]]["cond2"][tax[["gelisols"]]["pedon"]==id,] <- TRUE
    tax[["gelisols"]]["tax"][tax[["gelisols"]]["pedon"]==id,] <- TRUE

    return(tax)

  } else {

    tax[["gelisols"]]["cond2"][tax[["gelisols"]]["pedon"]==id,] <- FALSE
    tax[["gelisols"]]["tax"][tax[["gelisols"]]["pedon"]==id,] <- FALSE

    return(tax)

  }

}
```

```
################################################################################
# TODO:
# - review cond2a; edit to account for cind.,frag., pum. material within O hz
# - check that correct bulk density is used for moist db
histosols <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  org_top <- diag$orgmin$org_top
  org_bot <- diag$orgmin$org_bottom



  if(is.na(org_top) & is.na(org_bot)) {

    tax[["histosols"]]["cond2"][tax[["histosols"]]["pedon"]==id,] <- FALSE
    tax[["histosols"]]["tax"][tax[["histosols"]]["pedon"]==id,] <- FALSE

    return(tax)

  }




  ## cond1
  diag_hz <- c("densic","lithic","paralithic","duripan")
  top_contact <- diag_top_depth(diag_hz=diag_hz,diag=diag,pedon_id=id)

  if(is.na(top_contact)) {

    cond1_depth <- 60

  } else if(top_contact < 60) {

    cond1_depth <- min(top_contact)

  } else {

    cond1_depth <- 60

  }
```

```
  cond1 <- (sum(hz$hzthk[which(hz$andic == TRUE & hz$hzdepb <= cond1_depth)])
/ cond1_depth) < 0.6


  if(!cond1) {

    tax[["histosols"]]["cond1"][tax[["histosols"]]["pedon"]==id,] <- FALSE
    tax[["histosols"]]["tax"][tax[["histosols"]]["pedon"]==id,] <- FALSE

    return(tax)

  }


  ## cond2: must be 2a, 2b, 2c, OR 2d (2d1 OR 2d2)


  ## cond2a

  if(is.na(top_contact)) {

    tax[["histosols"]]["cond2a"][tax[["histosols"]]["pedon"]==id,] <- FALSE

  } else {

    if(any(hz$cindfragpum)) {

      cond2a <- (max(hz$hzdept[which(hz$cindfragpum==TRUE)]) == org_bot) &
        (min(hz$hzdept[which(hz$cindfragpum==TRUE)]) == min(top_contact))

    } else {

      cond2a <- FALSE

    }


    if(cond2a) {

      tax[["histosols"]]["cond2a"][tax[["histosols"]]["pedon"]==id,] <- TRUE
      tax[["histosols"]]["tax"][tax[["histosols"]]["pedon"]==id,] <- TRUE

      return(tax)

    }

  }



  ## cond2b

  hz_sl <- prof_slice(bottom=50,top=min(hz$hzdept),hz=hz)
  cond2b_org_top <- min(hz_sl$hzdept[which(hz_sl$desgnmaster == "O")])
```

```
  cond2b_org_bot <- max(hz_sl$hzdepb[which(hz_sl$desgnmaster == "O")])

  cond2b_und <- which(hz_sl$cindfragpum==TRUE & hz_sl$hzdept ==
cond2b_org_bot)

  cond2b <- ((cond2b_org_bot - cond2b_org_top) +
sum(hz_sl$hzthk[cond2b_und])) >= 40

  if(cond2b) {

    tax[["histosols"]]["cond2b"][tax[["histosols"]]["pedon"]==id,] <- TRUE
    tax[["histosols"]]["tax"][tax[["histosols"]]["pedon"]==id,] <- TRUE

    return(tax)

  }

  tax[["histosols"]]["cond2b"][tax[["histosols"]]["pedon"]==id,] <- FALSE




  ## cond2c

  min_hz <- which(hz$desgnmaster != "O")
  min_hz_thk <- sum(hz$hzthk[min_hz])

  if(min_hz_thk > 10) {

    cond2c <- FALSE


  } else {

    if(is.na(top_contact)) {

      hz_sl_2c <- hz
    } else {

      hz_sl_2c <- prof_slice(bottom=min(top_contact),top=0,hz=hz)

    }


    tot_thk <- sum(hz_sl_2c$hzthk)
    org_thk <- sum(hz_sl_2c$hzthk[which(hz_sl_2c$desgnmaster == "O")])

    cond2c <- (org_thk / tot_thk) >= (2/3)

  }

  if(cond2c) {

    tax[["histosols"]]["cond2c"][tax[["histosols"]]["pedon"]==id,] <- TRUE
```

```
    tax[["histosols"]]["tax"][tax[["histosols"]]["pedon"]==id,] <- TRUE

    return(tax)

  }

  tax[["histosols"]]["cond2c"][tax[["histosols"]]["pedon"]==id,] <- FALSE




  ## cond2d


  cond2d_ub <- org_top <= 40

  ## cond2d thicknesses; cond2d1, cond2d2

  if(cond2d_ub) {

    org_thk <- org_bot - org_top

    org_hz <- which(hz$desgnmaster=="O")

    bd_wa <- weighted.mean(hz$dbthirdbar[org_hz],hz$hzthk[org_hz]/
sum(hz$hzthk[org_hz]))

    cond2d1 <- org_thk >= 60 & (sum(hz$moss_vol[org_hz]) >= 75 | bd_wa <= 0.1)

    if(cond2d1) {

      tax[["histosols"]]["cond2d1"][tax[["histosols"]]["pedon"]==id,] <- TRUE
      tax[["histosols"]]["cond2d"][tax[["histosols"]]["pedon"]==id,] <- TRUE
      tax[["histosols"]]["tax"][tax[["histosols"]]["pedon"]==id,] <- TRUE

    } else {


      sap_hem <- which(hz$desgnmaster=="O" &
grepl(hz$hzname,pattern="[AaEe]"))
      fib <- which(hz$desgnmaster=="O" & grepl(hz$hzname,pattern="[Ii]")
                   & hz$moss_vol < 75 & hz$dbthirdbar >= 0.1)

      sap_hem_thk <- sum(hz$hzthk[sap_hem])
      fib_thk <- sum(hz$hzthk[fib])

      cond2d2 <- (sap_hem_thk + fib_thk) >= 40

      if(cond2d2) {
```

```
        tax[["histosols"]]["cond2d2"][tax[["histosols"]]["pedon"]==id,] <-
TRUE
        tax[["histosols"]]["cond2d"][tax[["histosols"]]["pedon"]==id,] <- TRUE
        tax[["histosols"]]["tax"][tax[["histosols"]]["pedon"]==id,] <- TRUE

      }

    }


  } else {

    tax[["histosols"]]["cond2d"][tax[["histosols"]]["pedon"]==id,] <- FALSE
    tax[["histosols"]]["cond2d1"][tax[["histosols"]]["pedon"]==id,] <- FALSE
    tax[["histosols"]]["cond2d2"][tax[["histosols"]]["pedon"]==id,] <- FALSE
    tax[["histosols"]]["tax"][tax[["histosols"]]["pedon"]==id,] <- FALSE

  }


  return(tax)


}









################################################################################
# TODO:
# - review texture interpretation
# - review albic hz interpretation
spodosols <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})



  ## cond0; check if argillic and kandic not above spodic
  arg_kan <- c("argillic","kandic")
  arg_kan_bot <- min(diag_top_depth(diag_hz=arg_kan,diag=diag,pedon_id=id))

  spodic_hz <- diag$spodic

  if(spodic_hz$hz) {

    if(is.na(arg_kan_bot)) {
```

```
      cond0 <- TRUE

    } else {

      cond0 <- arg_kan_bot < spodic_hz$bottom_depth

    }

  } else {

    cond0 <- TRUE

  }

  if(!cond0) {

    tax[["spodosols"]]["cond0"][tax[["spodosols"]]["pedon"]==id,] <- FALSE
    tax[["spodosols"]]["tax"][tax[["spodosols"]]["pedon"]==id,] <- FALSE

    return(tax)

  }



  ## cond1

  cond1 <- any(spodic_hz$hz & diag$albic$hz & diag$str$str %in%
c("gelic","cryic"))

  if(cond1) {

    tax[["spodosols"]]["cond1"][tax[["spodosols"]]["pedon"]==id,] <- TRUE
    tax[["spodosols"]]["tax"][tax[["spodosols"]]["pedon"]==id,] <- TRUE

    return(tax)

  }


  ## cond2

  cond2 <- which(hz$desgnmaster == "Ap" & hz$spodic_mat_pct > 85)

  if(length(cond2)>0) {

    tax[["spodosols"]]["cond2"][tax[["spodosols"]]["pedon"]==id,] <- TRUE
    tax[["spodosols"]]["tax"][tax[["spodosols"]]["pedon"]==id,] <- TRUE

    return(tax)

  }
```

```
  ## cond3

  if(!spodic_hz$hz) {

    tax[["spodosols"]]["cond3"][tax[["spodosols"]]["pedon"]==id,] <- FALSE
    tax[["spodosols"]]["tax"][tax[["spodosols"]]["pedon"]==id,] <- FALSE

    return(tax)

  } else {

    tax[["spodosols"]]["cond3"][tax[["spodosols"]]["pedon"]==id,] <- TRUE

  }


  ## cond3a

  cond3a1 <- spodic_hz$thick >= 10
  cond3a2 <- any(hz$desgnmaster == "Ap" & hz$hzdepb >= spodic_hz$top_depth)

  spod_sl <- prof_slice(spodic_hz$bottom_depth,spodic_hz$top_depth,hz)

  cond3a3 <- (sum(spod_sl$hzthk[spod_sl$cemented]) / sum(spod_sl$hzthk)) >=
0.5
  cond3a4 <- any(spod_sl$texture %in% c("clay","silty clay","sandy
clay","silty clay loam",
                                        "clay loam","sandy clay
loam","silt","silt loam",
                                        "loam","very fine sandy loam","fine
sandy loam",
                                        "sandy loam","coarse sandy
loam","loamy very fine sand",
                                        "loamy fine sand","loamy sand","loamy
coarse sand",
                                        "very fine sand","fine sand","sand") &
                   diag$str$str == "frigid")
  cond3a5 <- any(diag$str$str %in% c("cryic","gelic"))

  tax[["spodosols"]]["cond3a1"][tax[["spodosols"]]["pedon"]==id,] <- cond3a1
  tax[["spodosols"]]["cond3a2"][tax[["spodosols"]]["pedon"]==id,] <- cond3a2
  tax[["spodosols"]]["cond3a3"][tax[["spodosols"]]["pedon"]==id,] <- cond3a3
  tax[["spodosols"]]["cond3a4"][tax[["spodosols"]]["pedon"]==id,] <- cond3a4
  tax[["spodosols"]]["cond3a5"][tax[["spodosols"]]["pedon"]==id,] <- cond3a5

  if(any(cond3a1,cond3a2,cond3a3,cond3a4,cond3a5)) {

    cond3a <- TRUE
    tax[["spodosols"]]["cond3a"][tax[["spodosols"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["spodosols"]]["cond3a"][tax[["spodosols"]]["pedon"]==id,] <- FALSE
```

```
    tax[["spodosols"]]["tax"][tax[["spodosols"]]["pedon"]==id,] <- FALSE

    return(tax)

  }



  ## cond3b

  minsurf <- diag$orgmin$min_top
  cond3b1 <- spodic_hz$top_depth <= (minsurf+50)
  cond3b2 <- spodic_hz$top_depth <= (minsurf+200) &
             any(hz$texture %in% c("coarse sand","sand","find sand","loamy
coarse sand",
                                   "loamy sand","loamy fine sand") &
                 hz$hzdepb <= spodic_hz$top_depth)

  tax[["spodosols"]]["cond3b1"][tax[["spodosols"]]["pedon"]==id,] <- cond3b1
  tax[["spodosols"]]["cond3b2"][tax[["spodosols"]]["pedon"]==id,] <- cond3b2

  if(any(cond3b1,cond3b2)) {

    cond3b <- TRUE
    tax[["spodosols"]]["cond3b"][tax[["spodosols"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["spodosols"]]["cond3b"][tax[["spodosols"]]["pedon"]==id,] <- FALSE
    tax[["spodosols"]]["tax"][tax[["spodosols"]]["pedon"]==id,] <- FALSE

    return(tax)

  }



  ## cond3c

  # skipped check if duripan, fragipan, densic, lithic, paralithic,
petroferric contact
  # assumes any of those contacts will end the spodic hz anyway
  cond3c1 <- spodic_hz$bottom_depth >= (minsurf+25)
  cond3c2a <- any(spod_sl$texture %in% c("clay","silty clay","sandy
clay","silty clay loam",
                                         "clay loam","sandy clay
loam","silt","silt loam",
                                         "loam","very fine sandy loam","fine sandy
loam",
                                         "sandy loam","coarse sandy loam","loamy
very fine sand",
                                         "loamy fine sand","loamy sand","loamy
coarse sand",
                                         "very fine sand","fine sand","sand") &
```

```
                    diag$str$str == "frigid")
  cond3c2b <- any(diag$str$str %in% c("cryic","gelic"))

  tax[["spodosols"]]["cond3c1"][tax[["spodosols"]]["pedon"]==id,] <- cond3c1
  tax[["spodosols"]]["cond3c2a"][tax[["spodosols"]]["pedon"]==id,] <- cond3c2a
  tax[["spodosols"]]["cond3c2b"][tax[["spodosols"]]["pedon"]==id,] <- cond3c2b


  if(any(cond3c2a,cond3c2b)) {

    cond3c2 <- TRUE
    tax[["spodosols"]]["cond3c2"][tax[["spodosols"]]["pedon"]==id,] <- TRUE

  } else {

    cond3c2 <- FALSE
    tax[["spodosols"]]["cond3c2"][tax[["spodosols"]]["pedon"]==id,] <- FALSE

  }


  if(any(cond3c1,cond3c2)) {

    cond3c <- TRUE
    tax[["spodosols"]]["cond3c"][tax[["spodosols"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["spodosols"]]["cond3c"][tax[["spodosols"]]["pedon"]==id,] <- FALSE
    tax[["spodosols"]]["tax"][tax[["spodosols"]]["pedon"]==id,] <- FALSE

    return(tax)

  }



  ## cond3d

  cond3d1 <- diag$albic$bottom_depth <= spodic_hz$top_depth


  diag_top <-
min(diag_top_depth(c("densic","lithic","paralithic","duripan","petrocalcic"),diag,id))


  org_andic <- hz$hzdept[which(hz$andic & hz$desgnmaster == "O")]
  surf <- min(c(minsurf,org_andic))

  if(is.na(diag_top) | diag_top > (surf+60)) {

    cond3d2_sl <- prof_slice(surf+60,0,hz)
```

```
    cond3d2a <- any(sum(cond3d2_sl$hzthk[which(cond3d2_sl$andic)])/
sum(cond3d2_sl$hzthk)<0.6)


  } else {

    cond3d2a <- FALSE

  }


  if(!is.na(diag_top)) {

    cond3d2_sl <- prof_slice(surf+diag_top,0,hz)
    cond3d2b <- any(sum(cond3d2_sl$hzthk[which(cond3d2_sl$andic)])/
sum(cond3d2_sl$hzthk)<0.6)

  } else {

    cond3d2b <- FALSE

  }



  tax[["spodosols"]]["cond3d1"][tax[["spodosols"]]["pedon"]==id,] <- cond3d1
  tax[["spodosols"]]["cond3d2a"][tax[["spodosols"]]["pedon"]==id,] <- cond3d2a
  tax[["spodosols"]]["cond3d2b"][tax[["spodosols"]]["pedon"]==id,] <- cond3d2b


  if(any(cond3d2a,cond3d2b)) {

    cond3d2 <- TRUE
    tax[["spodosols"]]["cond3d2"][tax[["spodosols"]]["pedon"]==id,] <- TRUE

  }

  if(any(cond3d1,cond3d2)) {

    cond3d <- TRUE
    tax[["spodosols"]]["cond3d"][tax[["spodosols"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["spodosols"]]["cond3d"][tax[["spodosols"]]["pedon"]==id,] <- FALSE
    tax[["spodosols"]]["tax"][tax[["spodosols"]]["pedon"]==id,] <- FALSE

    return(tax)

  }


  ## sanity check
  if(any(cond3a,cond3b,cond3c,cond3d)) {
```

```
    tax[["spodosols"]]["tax"][tax[["spodosols"]]["pedon"]==id,] <- TRUE

  }


  return(tax)

}









################################################################################
andisols <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})



  diag_top <-
min(diag_top_depth(c("densic","lithic","paralithic","duripan","petrocalcic"),diag,id))

  minsurf <- diag$orgmin$min_top


  org_andic <- hz$hzdept[which(hz$andic & hz$desgnmaster == "O")]
  surf <- min(c(minsurf,org_andic))

  if(is.na(diag_top) | diag_top > (surf+60)) {

    sl <- prof_slice(surf+60,min(hz$hzdept),hz)

    cond1 <- any(sum(sl$hzthk[which(sl$andic)])/sum(sl$hzthk)>0.6)
    cond2 <- FALSE

  } else {

    cond1 <- FALSE
    sl <- prof_slice(surf+diag_top,min(hz$hzdept),hz)
    cond2 <- any(sum(sl$hzthk[which(sl$andic)])/sum(sl$hzthk)>0.6)

    tax[["andisols"]]["cond2"][tax[["andisols"]]["pedon"]==id,] <- cond2

  }
```

```
  tax[["andisols"]]["cond1"][tax[["andisols"]]["pedon"]==id,] <- cond1


  if(any(cond1,cond2)) {

    tax[["andisols"]]["tax"][tax[["andisols"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["andisols"]]["tax"][tax[["andisols"]]["pedon"]==id,] <- FALSE

  }


  return(tax)


}









################################################################################
# TODO:
# - review cond2 interpretation
oxisols <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})


  minsurf <- diag$orgmin$min_top


  ## cond1

  if(diag$oxic$hz) {

    cond1 <- (diag$oxic$top_depth <= minsurf+150) &
             (diag$kandic$hz==FALSE | diag$kandic$top_depth >= minsurf+150)
    cond2 <- NA

  } else {

    cond1 <- FALSE
```

```
    if(diag$kandic$hz & diag$kandic$top_depth <= minsurf+100) {

      sl <- prof_slice(minsurf+18,minsurf,hz)
      cond2 <- weighted.mean(sl$claytotal,sl$hzthk/sum(sl$hzthk)) >= 40

    } else {

      cond2 <- FALSE

    }

  }


  tax[["oxisols"]]["cond1"][tax[["oxisols"]]["pedon"]==id,] <- cond1
  tax[["oxisols"]]["cond2"][tax[["oxisols"]]["pedon"]==id,] <- cond2


  if(any(cond1,cond2)) {

    tax[["oxisols"]]["tax"][tax[["oxisols"]]["pedon"]==id,] <- TRUE

  }


  return(tax)


}









################################################################################
# TODO:
# - add cond1 ped angles?
# - cond3 cracks
vertisols <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})


  minsurf <- diag$orgmin$min_top

  ## cond1
  ## angles not checked
```

```
  cond1 <- any(hz$hzthk>=25 & hz$hzdept <= minsurf+100 &
           (grepl(hz$hzname,pattern="[SSss]") | hz$structtype=="wedge"))


  if(!cond1) {

    tax[["vertisols"]]["cond1"][tax[["vertisols"]]["pedon"]==id,] <- FALSE
    tax[["vertisols"]]["tax"][tax[["vertisols"]]["pedon"]==id,] <- FALSE

    return(tax)

  } else {

    tax[["vertisols"]]["cond1"][tax[["vertisols"]]["pedon"]==id,] <- TRUE

  }



  ## cond2


  diag_top <-
diag_top_depth(c("densic","lithic","paralithic","duripan","petrocalcic"),diag,id)

  if(!all(is.na(diag_top))) {

    diag_top <-
min(diag_top_depth(c("densic","lithic","paralithic","duripan","petrocalcic"),diag,id),na.r

  }

  cond2_thicker <- max(c(minsurf+18, hz$hzthk[which(hz$hzname=="Ap")]))

  sl1 <- prof_slice(cond2_thicker, minsurf, hz)
  sl1_clay <- weighted.mean(sl1$claytotal,sl1$hzthk/sum(sl1$hzthk)) >= 30 #
logical

  if(is.na(diag_top)) {

    sl2 <- prof_slice(50, 18, hz)

  } else {

    sl2 <- prof_slice(min(c(50,diag_top)), 18, hz)

  }

  sl2_clay <- weighted.mean(sl2$claytotal,sl2$hzthk/sum(sl2$hzthk)) >= 30 #
logical


  if(sl1_clay & sl2_clay) {
```

```
    cond2 <- TRUE
    tax[["vertisols"]]["cond2"][tax[["vertisols"]]["pedon"]==id,] <- TRUE

  } else {

    cond2 <- FALSE
    tax[["vertisols"]]["cond2"][tax[["vertisols"]]["pedon"]==id,] <- FALSE
    tax[["vertisols"]]["tax"][tax[["vertisols"]]["pedon"]==id,] <- FALSE

    return(tax)

  }

  ## sanity check

  if(cond1 & cond2) {

    tax[["vertisols"]]["tax"][tax[["vertisols"]]["pedon"]==id,] <- TRUE

    return(tax)

  }



}









################################################################################
# TODO:
# - review cond2a, cond2b
aridisols <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})


  ## cond1 (a AND b AND (c OR d)) OR cond2 (a AND b AND c)

  ## cond1

  cond1a <- diag$smr$smr == "aridic"
  cond1b <- diag$epipedon$hz %in% c("ochric","anthropic")

  tax[["aridisols"]]["cond1a"][tax[["aridisols"]]["pedon"]==id,] <- cond1a
```

```
  tax[["aridisols"]]["cond1b"][tax[["aridisols"]]["pedon"]==id,] <- cond1b


  if(!any(cond1a,cond1b)) {

    tax[["aridisols"]]["cond1"][tax[["aridisols"]]["pedon"]==id,] <- FALSE

  }


  ## cond1c

  if(diag$cambic$hz) {

    cond1c <- (diag$cambic$top_depth <= 100 & diag$cambic$bottom_depth >= 25)
|
      (diag$str$str=="cryic" & diag$cambic$hz==TRUE) |
      any(diag$anhydric$hz, diag$calcic$hz, diag$gypsic$hz,
diag$petrocalcic$hz,
          diag$petrogypsic$hz, diag$salic$hz, diag$duripan$hz)

  } else {

    cond1c <- (diag$str$str=="cryic" & diag$cambic$hz==TRUE) |
      any(diag$anhydric$hz, diag$calcic$hz, diag$gypsic$hz,
diag$petrocalcic$hz,
          diag$petrogypsic$hz, diag$salic$hz, diag$duripan$hz)
  }


  tax[["aridisols"]]["cond1c"][tax[["aridisols"]]["pedon"]==id,] <- cond1c


  ## cond1d

  cond1d <- any(diag$argillic$hz, diag$natric$hz)

  tax[["aridisols"]]["cond1d"][tax[["aridisols"]]["pedon"]==id,] <- FALSE

  if(cond1a & cond1b & any(cond1c,cond1d)) {

    tax[["aridisols"]]["cond1"][tax[["aridisols"]]["pedon"]==id,] <- TRUE
    tax[["aridisols"]]["tax"][tax[["aridisols"]]["pedon"]==id,] <- TRUE

    return(tax)

  } else {

    tax[["aridisols"]]["cond1"][tax[["aridisols"]]["pedon"]==id,] <- FALSE
    tax[["aridisols"]]["tax"][tax[["aridisols"]]["pedon"]==id,] <- FALSE

  }
```

```
  ## cond2
  ## skipping cond2a and cond2b

  cond2 <- diag$salic$hz

  if(diag$sulfuric$hz) {

    cond2c <- diag$sulfuric$top_depth >= 150

  } else {

    cond2c <- TRUE

  }


  tax[["aridisols"]]["cond2"][tax[["aridisols"]]["pedon"]==id,] <- cond2
  tax[["aridisols"]]["cond2c"][tax[["aridisols"]]["pedon"]==id,] <- cond2c


  if(cond2 & cond2c) {

    tax[["aridisols"]]["tax"][tax[["aridisols"]]["pedon"]==id,] <- TRUE

    return(tax)

  } else {

    tax[["aridisols"]]["tax"][tax[["aridisols"]]["pedon"]==id,] <- FALSE

    return(tax)

  }



}









################################################################################
# TODO:
# - review depth interpretations
# - if pedon is not deep info not deep enough for check range, select deepest
horizon
ultisols <- function(hzs, diags, tax, id) {
```

```
  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})


  ## cond1 (a OR b) OR cond2 (a AND b)

  minsurf <- diag$orgmin$min_top
  diag_tops <-
min(diag_top_depth(c("densic","lithic","paralithic","petroferric"),diag,id))

  cond1 <- (diag$argillic$hz | diag$kandic$hz) & !diag$fragipan$hz

  if(cond1) {

    ## if cond1, check 1a and 1b
    ## else, go to cond2

    epi_sl <-
prof_slice(diag$epipedon$bottom_depth,diag$epipedon$top_depth,hz)
    cond1a <- epi_sl$texture %in% c("coarse sand","sand","fine sand","loamy
coarse sand,
                                    loamy sand","loamy fine sand")

    if(diag$argillic$hz) {

      cond1a1_ub <- diag$argillic$top_depth

      if(cond1a1_ub+125 >= 200) {

        cond1a1_ub <- 200

      } else {

        cond1a1_ub <- cond1a1_ub+125

      }

      cond1a1_deeper <- max(cond1a1_ub,minsurf+180)

      if(cond1a1_deeper > max(hz$hzdepb)) {
        cond1a1 <- any(hz$bs82[nrow(hz)] < 35 ,na.rm=TRUE)
      } else {
        cond1a1 <- any(hz$bs82 < 35 & (hz$hzdept <= cond1a1_deeper & hz$hzdepb
>= cond1a1_deeper),na.rm=TRUE)
      }




      if(is.na(diag_tops)) {

        cond1a2 <- FALSE

      } else if(diag_tops < cond1a1_deeper) {
```

```
        cond1a2 <- any(hz$bs82 < 35 & (hz$hzdept <= diag_tops & hz$hzdepb >=
diag_tops),na.rm=TRUE)

      } else {

        cond1a2 <- FALSE

      }


      tax[["ultisols"]]["cond1a1"][tax[["ultisols"]]["pedon"]==id,] <- cond1a1
      tax[["ultisols"]]["cond1a2"][tax[["ultisols"]]["pedon"]==id,] <- cond1a2


      if(any(cond1a1,cond1a2)) {

        cond1a <- TRUE
        tax[["ultisols"]]["cond1"][tax[["ultisols"]]["pedon"]==id,] <- TRUE
        tax[["ultisols"]]["cond1a"][tax[["ultisols"]]["pedon"]==id,] <- TRUE
        tax[["ultisols"]]["tax"][tax[["ultisols"]]["pedon"]==id,] <- TRUE

        return(tax)


      } else {

        cond1a <- FALSE
        tax[["ultisols"]]["cond1a"][tax[["ultisols"]]["pedon"]==id,] <- FALSE

      }

    }




    ## cond1b

    cond1b1_depth <-
min(diag$argillic$top_depth,diag$kandic$top_depth,na.rm=TRUE) + 125
    cond1b2_depth <- minsurf + 180

    if(is.na(diag_tops)) {

      cond1b3_depth <- max(cond1b1_depth,cond1b2_depth) + 1 # make sure it is
never chosen

    } else {

      cond1b3_depth <- diag_tops

    }
```

```
    if(cond1b1_depth == min(cond1b1_depth,cond1b2_depth,cond1b3_depth)) {

      cond1b_shallowest <- cond1b1_depth

      if(cond1b_shallowest > max(hz$hzdepb)) {
        cond1b1 <- any(hz$bs82[nrow(hz)] < 35 ,na.rm=TRUE)
      } else {
        cond1b1 <- any(hz$bs82 < 35 & (hz$hzdept <= cond1b_shallowest &
hz$hzdepb >= cond1b_shallowest),na.rm=TRUE)
      }

      cond1b2 <- FALSE
      cond1b3 <- FALSE

    } else if(cond1b2_depth == min(cond1b1_depth,cond1b2_depth,cond1b3_depth))
{

      cond1b_shallowest <- cond1b2_depth
      cond1b1 <- FALSE

      if(cond1b_shallowest > max(hz$hzdepb)) {
        cond1b2 <- any(hz$bs82[nrow(hz)] < 35 ,na.rm=TRUE)
      } else {
        cond1b2 <- any(hz$bs82 < 35 & (hz$hzdept <= cond1b_shallowest &
hz$hzdepb >= cond1b_shallowest),na.rm=TRUE)
      }

      cond1b3 <- FALSE

    } else if(cond1b3_depth == min(cond1b1_depth,cond1b2_depth,cond1b3_depth))
{

      cond1b_shallowest <- cond1b3_depth
      cond1b1 <- FALSE
      cond1b2 <- FALSE

      if(cond1b_shallowest > max(hz$hzdepb)) {
        cond1b3 <- any(hz$bs82[nrow(hz)] < 35 ,na.rm=TRUE)
      } else {
        cond1b3 <- any(hz$bs82 < 35 & (hz$hzdept <= cond1b_shallowest &
hz$hzdepb >= cond1b_shallowest),na.rm=TRUE)
      }


    }

    tax[["ultisols"]]["cond1b1"][tax[["ultisols"]]["pedon"]==id,] <- cond1b1
    tax[["ultisols"]]["cond1b2"][tax[["ultisols"]]["pedon"]==id,] <- cond1b2
    tax[["ultisols"]]["cond1b3"][tax[["ultisols"]]["pedon"]==id,] <- cond1b3



    if(any(cond1b1,cond1b2,cond1b3)) {
```

```
      cond1b <- TRUE
      tax[["ultisols"]]["cond1b"][tax[["ultisols"]]["pedon"]==id,] <- TRUE

      return(tax)

    } else {

      cond1b <- FALSE
      tax[["ultisols"]]["cond1b"][tax[["ultisols"]]["pedon"]==id,] <- FALSE

    }

    if(cond1a==FALSE & cond1b==FALSE) {

      tax[["ultisols"]]["cond1"][tax[["ultisols"]]["pedon"]==id,] <- FALSE

    }


  } else {

    tax[["ultisols"]]["cond1"][tax[["ultisols"]]["pedon"]==id,] <- FALSE

  }


  ## end of cond1


  ## cond2

  if(diag$fragipan$hz) {

    cond2a <- diag$argillic$hz | diag$kandic$hz


    cond2b1_depth <- diag$fragipan$top_depth + 75
    cond2b2_depth <- minsurf + 200

    if(is.na(diag_tops)) {

      cond2b3_depth <- max(cond2b1_depth,cond2b2_depth) + 1 # make sure it is
never chosen

    } else {

      cond2b3_depth <- diag_tops

    }

    if(cond2b1_depth == min(cond2b1_depth,cond2b2_depth,cond2b3_depth)) {

      cond2b_shallowest <- cond2b1_depth
```

```
      if(cond2b_shallowest > max(hz$hzdepb)) {
        cond2b1 <- any(hz$bs82[nrow(hz)] < 35, na.rm=TRUE)
      } else {
        cond2b1 <- any(hz$bs82 < 35 & (hz$hzdept <= cond2b_shallowest &
hz$hzdepb >= cond2b_shallowest),na.rm=TRUE)
      }

      cond2b2 <- FALSE
      cond2b3 <- FALSE

    } else if(cond2b2_depth == min(cond2b1_depth,cond2b2_depth,cond2b3_depth))
{

      cond2b_shallowest <- cond2b2_depth
      cond2b1 <- FALSE

      if(cond2b_shallowest > max(hz$hzdepb)) {
        cond2b2 <- any(hz$bs82[nrow(hz)] < 35, na.rm=TRUE)
      } else {
        cond2b2 <- any(hz$bs82 < 35 & (hz$hzdept <= cond2b_shallowest &
hz$hzdepb >= cond2b_shallowest),na.rm=TRUE)
      }


      cond2b3 <- FALSE

    } else if(cond2b3_depth == min(cond2b1_depth,cond2b2_depth,cond2b3_depth))
{

      cond2b_shallowest <- cond2b3_depth
      cond2b1 <- FALSE
      cond2b2 <- FALSE

      if(cond2b_shallowest > max(hz$hzdepb)) {
        cond2b3 <- any(hz$bs82[nrow(hz)] < 35, na.rm=TRUE)
      } else {
        cond2b3 <- any(hz$bs82 < 35 & (hz$hzdept <= cond2b_shallowest &
hz$hzdepb >= cond2b_shallowest),na.rm=TRUE)
      }


    }


    tax[["ultisols"]]["cond2b1"][tax[["ultisols"]]["pedon"]==id,] <- cond2b1
    tax[["ultisols"]]["cond2b2"][tax[["ultisols"]]["pedon"]==id,] <- cond2b2
    tax[["ultisols"]]["cond2b3"][tax[["ultisols"]]["pedon"]==id,] <- cond2b3



    if(any(cond2b1,cond2b2,cond2b3)) {

      cond2b <- TRUE
```

```
      tax[["ultisols"]]["cond2b"][tax[["ultisols"]]["pedon"]==id,] <- TRUE


    } else {

      cond2b <- FALSE
      tax[["ultisols"]]["cond2b"][tax[["ultisols"]]["pedon"]==id,] <- FALSE

    }

    if(cond2a & cond2b) {

      tax[["ultisols"]]["cond2"][tax[["ultisols"]]["pedon"]==id,] <- TRUE
      tax[["ultisols"]]["tax"][tax[["ultisols"]]["pedon"]==id,] <- TRUE

    } else {

      tax[["ultisols"]]["cond2"][tax[["ultisols"]]["pedon"]==id,] <- FALSE
      tax[["ultisols"]]["tax"][tax[["ultisols"]]["pedon"]==id,] <- FALSE

    }


  } else {

    tax[["ultisols"]]["cond2"][tax[["ultisols"]]["pedon"]==id,] <- FALSE

  }


  return(tax)



}









################################################################################
# TODO:
# - update mollic epipedon thickness check if epi edited
# - mollic epipedon must be checked before checking this
# - verifying correct base saturation
mollisols <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
```

```
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})


  ## cond1 (a OR b) AND cond2

  cond1a <- diag$mollic$hz

  if(!cond1a) {

    tax[["mollisols"]]["cond1a"][tax[["mollisols"]]["pedon"]==id,] <- FALSE

    mollic <- diag$mollic

    # true if thk req NOT met
    idx <-
which(mollic[,c("cond6a1","cond6a2","cond6a3","cond6c1","cond6c2","cond8")]==FALSE)
    if(length(idx)==0) {

      cond1b_moll_thk <- FALSE # if all NA, then thickness not checked; i.e.
other conds not met

    } else {

      cond1b_moll_thk <- !
any(mollic[,c("cond6a1","cond6a2","cond6a3","cond6c1","cond6c2","cond8")
[idx]])

    }


    cond1b_diag_reqs1 <- cond1b_moll_thk &
any(diag$argillic$hz,diag$kandic$hz,diag$natric$hz) & diag$albic$hz

    arg_kan_nat_tops <-
diag_top_depth(c("argillic","kandic","natric"),diag,id)

    if(!is.na(arg_kan_nat_tops)) {

      arg_kan_nat_tops <-
min(diag_top_depth(c("argillic","kandic","natric"),diag,id))

    }

    if(is.na(arg_kan_nat_tops)) {

      cond1b_diag_reqs1 <- FALSE
      cond1b_diag_reqs2 <- FALSE

    } else if(cond1b_diag_reqs1) {

      # check if albic hz above argillic,kandic,natric in order to separate
from surface
      cond1b_diag_reqs2 <- diag$albic$bottom_depth < arg_kan_nat_tops
```

```
    } else {

      cond1b_diag_reqs2 <- FALSE

    }


    if(cond1b_diag_reqs1 & cond1b_diag_reqs2) {

      ## check mollic structure, color, OC, and BS

      arg_kan_nat_bots <-
max(diag_bot_depth(c("argillic","kandic","natric"),diag,id))

      sl <- prof_slice(arg_kan_nat_bots,arg_kan_nat_tops,hz)


      ## mollic structure

      mollic_cond1a <- which(
        tolower(sl$structtype) %in% c("granular","platy") |
          (tolower(sl$structtype) %in% c("lenticular","angular
blocky","subangular blocky") & sl$structsize != "very coarse") |
          (tolower(sl$structtype) %in% c("prismatic","columnar","wedge") &
sl$structsize %in% c("very fine","fine","medium"))
      )


      mollic_cond1b <- which(tolower(sl$rrc) %in% c("moderately
mard","slightly Hard","soft","loose",
                                                    "extremely weak","very
weak","weak","moderate",
                                                    "mh","sh","s","l","ew","vw","w","m"))


      mollic_cond1 <- list(mollic_cond1a,mollic_cond1b)
[which.max(lengths(list(mollic_cond1a,mollic_cond1b)))]


      ## mollic strucutre - rock frags

      mollic_cond2 <- mollic_cond1[mollic_cond1 %in% which(sl$fragtotvol <
50)]

      if(length(mollic_cond2)==0) {

        mollic_structure <- FALSE

      } else {

        mollic_strucutre <- TRUE

      }
```

```
      ## mollic color

      ## Cond 3: both 3a1 and 3a2 OR 3b OR 3c

      mollic_cond3a1 <- mollic_cond2[mollic_cond2 %in% which(sl$colorvalue_w
<= 3 & sl$colorvalue_d <= 5)]
      mollic_cond3a2 <- mollic_cond3a1[mollic_cond3a1 %in%
which(sl$colorchroma_w <= 3)]

      mollic_cond3b <- mollic_cond2[mollic_cond2 %in%
which((sl$claysizedcarb[mollic_cond2] >= 15 & sl$claysizedcarb[mollic_cond2]
<= 40) &
                                          (sl$colorvalue_w[mollic_cond2] <= 3 &
sl$colorchroma_w[mollic_cond2] <= 3))]

      mollic_cond3c <- mollic_cond2[mollic_cond2 %in%
which((sl$claysizedcarb[mollic_cond2] >= 40) &
                                          (sl$colorvalue_w[mollic_cond2] <=
5))]

      mollic_cond3 <- list(mollic_cond3a2,mollic_cond3b,mollic_cond3c)
      mollic_cond3 <- mollic_cond3[which.max(lapply(mollic_cond3,length))]

      if(length(mollic_cond3)==0) {

        mollic_color <- FALSE

      } else {

        mollic_color <- TRUE

      }



      ## mollic base sat

      mollic_cond4 <- mollic_cond3[mollic_cond3 %in% which(sl$bs82 >= 50)]

      if(length(mollic_cond4)==0) {

        mollic_basesat <- FALSE

      } else {

        mollic_basesat <- TRUE

      }
```

```
      ## mollic OC


      mollic_cond5a_colorval <- mollic_cond4[mollic_cond4 %in%
which(sl$colorvalue_w %in% c(4,5))]

      mollic_cond5a <- mollic_cond5a_colorval[mollic_cond5a_colorval %in%
which(sl$oc >= 2.5)]


      ## Cond 5b

      if(any("C" %in% sl$desgnmaster)) {

        c_idx <- which.min(sl$desgnmaster == "C")

        c_value_w <- sl$colorvalue_w[c_idx]
        c_value_d <- sl$colorvalue_d[c_idx]

        c_chroma_w <- sl$colorchroma_w[c_idx]
        c_chroma_d <- sl$colorchroma_d[c_idx]

        c_oc <- sl$oc[c_idx]

        mollic_cond5b_val_w <- mollic_cond4[mollic_cond4 %in%
which(sl$colorvalue_w < c_value_w-1)]

        mollic_cond5b_chroma_w <- mollic_cond4[mollic_cond4 %in%
which(sl$colorchroma_w < c_chroma_w-2)]
        mollic_cond5b_chroma_d <- mollic_cond4[mollic_cond4 %in%
which(sl$colorchroma_d < c_chroma_d-2)]

        mollic_cond5b_chroma <-
list(mollic_cond5b_chroma_w,mollic_cond5b_chroma_d)
        mollic_cond5b_chroma <-
mollic_cond5b_chroma[which.max(lapply(mollic_cond5b_chroma,length))]


        mollic_cond5b_idx <- list(mollic_cond5b_val_w,mollic_cond5b_chroma)
        mollic_cond5b_idx <-
mollic_cond5b_idx[which.max(lapply(mollic_cond5b_idx,length))]


        mollic_cond5b <- mollic_cond5b_idx[mollic_cond5b_idx %in% which(sl$oc
> 1.006*c_oc)]

      }

      ## Cond 5c



      mollic_cond5c <- mollic_cond4[mollic_cond4 %in% which(sl$oc >= 0.6)]
```

```
    mollic_cond5 <- list(mollic_cond5a,mollic_cond5b,mollic_cond5c)
    mollic_cond5 <- mollic_cond5[which.max(lapply(mollic_cond5,length))]


    if(length(mollic_cond5)==0) {

      mollic_oc <- FALSE

    } else {

      cond1b_thk <- sum(sl$hzthk[mollic_cond5])

      if(cond1b_thk) {

        mollic_oc <- TRUE

      } else {

        mollic_oc <- FALSE

      }

    }


    ## sanity check

    if(all(mollic_structure,mollic_color,mollic_basesat,mollic_oc)) {

      cond1b <- TRUE

    } else {

      cond1b <- FALSE

    }



  } else {

    cond1b <- FALSE

  }

  tax[["mollisols"]]["cond1b"][tax[["mollisols"]]["pedon"]==id,] <- cond1b


} else {

  cond1b <- TRUE
```

```
  tax[["mollisols"]]["cond1a"][tax[["mollisols"]]["pedon"]==id,] <- TRUE

}


if(any(cond1a,cond1b)) {

  cond1 <- TRUE

} else {

  cond1 <- FALSE

}



## cond2

minsurf <- diag$orgmin$min_top

arg_kan_nat_tops <- diag_top_depth(c("argillic","kandic","natric"),diag,id)
contacts <- diag_top_depth(c("densic","lithic","paralithic"),diag,id)

if(!all(is.na(arg_kan_nat_tops))) {

  arg_kan_nat <- min(arg_kan_nat_tops) + 125

} else {

  arg_kan_nat <- max(hz$hzdepb) + 1

}

if(!all(is.na(contacts))) {

  contacts <- min(contacts) + minsurf

} else {

  contacts <- max(hz$hzdepb) + 1

}


if(!all(is.na(arg_kan_nat_tops))) {

  tops <- c(arg_kan_nat_tops,minsurf,minsurf)
  bots <- c(arg_kan_nat,180,contacts)

  cond2_top <- tops[which.min(bots)] # this is not a typo
  cond2_bot <- bots[which.min(bots)]
```

```
    sl2 <- prof_slice(cond2_bot,cond2_top,hz)

    cond2 <- all(sl2$bs7 >= 50, na.rm=TRUE)

  } else {

    sl2 <- prof_slice(180,minsurf,hz)

    cond2 <- all(sl2$bs7 >= 50, na.rm=TRUE)

  }


  tax[["mollisols"]]["cond1"][tax[["mollisols"]]["pedon"]==id,] <- cond1
  tax[["mollisols"]]["cond2"][tax[["mollisols"]]["pedon"]==id,] <- cond2

  if(all(cond1,cond2)) {

    tax[["mollisols"]]["tax"][tax[["mollisols"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["mollisols"]]["tax"][tax[["mollisols"]]["pedon"]==id,] <- FALSE

  }


  return(tax)



}









################################################################################
# TODO:
# - plaggen not checked
alfisols <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})
```

```
  cond1 <- any(diag$argillic$hz,diag$kandic$hz,diag$natric$hz)

  if(diag$fragipan$hz) {

    sl <- prof_slice(diag$fragipan$bottom_depth,diag$fragipan$top_depth,hz)

    cond2 <- any(sl$clay_films)

  } else {

    cond2 <- FALSE

  }

  tax[["alfisols"]]["cond1"][tax[["alfisols"]]["pedon"]==id,] <- cond1
  tax[["alfisols"]]["cond2"][tax[["alfisols"]]["pedon"]==id,] <- cond2

  if(any(cond1,cond2)) {

    tax[["alfisols"]]["tax"][tax[["alfisols"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["alfisols"]]["tax"][tax[["alfisols"]]["pedon"]==id,] <- FALSE

  }


  return(tax)



}









################################################################################
# TODO:
# - cond2 sulfidic materials not checked
# - cond2b3 groundwater not checked
# - cond2b3 first hz checked for decreasing ESP/SAR
inceptisols <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})
```

```
  ## cond 1 (a OR b OR c OR d OR e) OR 2 (a AND b (1 OR 2 OR 3))


  ## cond1

  ## cond1a

  minsurf <- diag$orgmin$min_top

  if(diag$cambic$hz) {

    cond1a <- diag$cambic$top_depth <= minsurf+100 & diag$cambic$bottom_depth
>= minsurf+25

  } else {

    cond1a <- FALSE

  }


  ## cond1b

  diags_hz <- c("calcic","petrocalcic","gypsic","placic","duripan")

  if(any(as.logical(lapply(diag[diags_hz],function(x){x[["hz"]]
[which(x[["pedon"]]==id)]})))) {

    diags_tops <- diag_top_depth(diags_hz,diag,id)
    diag_surf <- diags_tops[which(!is.na(diags_tops))]

    cond1b <- diag_surf <= minsurf+100

  } else {

    cond1b <- FALSE

  }


  ## cond1c

  diags_hz <- c("fragipan","oxic","sombric","spodic")

  if(any(as.logical(lapply(diag[diags_hz],function(x){x[["hz"]]
[which(x[["pedon"]]==id)]})))) {

    diags_tops <- diag_top_depth(diags_hz,diag,id)
    diag_surf <- diags_tops[which(!is.na(diags_tops))]

    cond1b <- diag_surf <= minsurf+200

  } else {
```

```
    cond1c <- FALSE

  }


  ## cond1d

  diags_hz <- c("sulfuric")

  if(any(as.logical(lapply(diag[diags_hz],function(x){x[["hz"]]
[which(x[["pedon"]]==id)]})))) {

    diags_tops <- diag_top_depth(diags_hz,diag,id)
    diag_surf <- diags_tops[which(!is.na(diags_tops))]

    cond1d <- diag_surf <= minsurf+150

  } else {

    cond1d <- FALSE

  }


  ## cond1e

  cond1e <- diag$str$str %in% c("cryic","gelic") & diag$cambic$hz


  tax[["inceptisols"]]["cond1a"][tax[["inceptisols"]]["pedon"]==id,] <- cond1a
  tax[["inceptisols"]]["cond1b"][tax[["inceptisols"]]["pedon"]==id,] <- cond1b
  tax[["inceptisols"]]["cond1c"][tax[["inceptisols"]]["pedon"]==id,] <- cond1c
  tax[["inceptisols"]]["cond1d"][tax[["inceptisols"]]["pedon"]==id,] <- cond1d
  tax[["inceptisols"]]["cond1e"][tax[["inceptisols"]]["pedon"]==id,] <- cond1e

  if(any(cond1a,cond1b,cond1c,cond1d,cond1e)) {

    cond1 <- TRUE
    tax[["inceptisols"]]["cond1"][tax[["inceptisols"]]["pedon"]==id,] <- TRUE


  } else {

    cond1 <- FALSE
    tax[["inceptisols"]]["cond1"][tax[["inceptisols"]]["pedon"]==id,] <- FALSE

  }


  ## cond2
  ## sulfidic materials not checked

  cond2a_sl <- prof_slice(minsurf+50,minsurf+20,hz)
```

```
  if(any(!is.na(cond2a_sl$n))) {
    cond2a <- any(cond2a_sl$n <= 0.7 | cond2a_sl$claytotal <= 8)
  } else {
    cond2a <- any(cond2a_sl$claytotal <= 8)
  }

  if(!cond2a) {

    cond2 <- FALSE
    tax[["inceptisols"]]["cond2"][tax[["inceptisols"]]["pedon"]==id,] <- FALSE
    tax[["inceptisols"]]["cond2a"][tax[["inceptisols"]]["pedon"]==id,] <-
FALSE


  }

  diags_hz <- c("folistic","histic","mollic","umbric")
  cond2b1 <- any(as.logical(lapply(diag[diags_hz],function(x){x[["hz"]]
[which(x[["pedon"]]==id)]})))

  cond2b2 <- diag$salic$hz

  if(max(hz$hzdepb) > 50) {
    sl1 <- prof_slice(minsurf+50,min(hz$hzdept),hz)
    sl2 <- prof_slice(max(hz$hzdepb),minsurf+50,hz)
    cond2b3 <- length(which(sl1$esp >= 15 | sl1$sar >=13)) >= nrow(sl1)/2 &
               (sl1$esp[nrow(sl1)] > sl2$esp[1] | sl1$sar[nrow(sl1)] >
sl2$sar[1])
  } else {

    sl1 <- prof_slice(minsurf+50,min(hz$hzdept),hz)
    cond2b3 <- length(which(sl1$esp >= 15 | sl1$sar >=13)) >= nrow(sl1)/2

  }


  tax[["inceptisols"]]["cond2b1"][tax[["inceptisols"]]["pedon"]==id,] <-
cond2b1
  tax[["inceptisols"]]["cond2b2"][tax[["inceptisols"]]["pedon"]==id,] <-
cond2b2
  tax[["inceptisols"]]["cond2b3"][tax[["inceptisols"]]["pedon"]==id,] <-
cond2b3


  if(any(cond2b1,cond2b2,cond2b3)) {

    cond2b <- TRUE

  } else {

    cond2b <- FALSE

  }
```

```
  tax[["inceptisols"]]["cond2a"][tax[["inceptisols"]]["pedon"]==id,] <- cond2a
  tax[["inceptisols"]]["cond2b"][tax[["inceptisols"]]["pedon"]==id,] <- cond2b

  if(all(cond2a,cond2b)) {

    cond2 <- TRUE
    tax[["inceptisols"]]["cond2"][tax[["inceptisols"]]["pedon"]==id,] <- TRUE

  } else {

    cond2 <- FALSE
    tax[["inceptisols"]]["cond2"][tax[["inceptisols"]]["pedon"]==id,] <- FALSE

  }


  if(any(cond1,cond2)) {

    tax[["inceptisols"]]["tax"][tax[["inceptisols"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["inceptisols"]]["tax"][tax[["inceptisols"]]["pedon"]==id,] <- FALSE

  }


  return(tax)

}









################################################################################
entisols <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  orders <- c("gelisols","histosols","spodosols","andisols","oxisols",
              "vertisols","aridisols","ultisols","mollisols","alfisols",
              "inceptisols")
```

```
  if(any(as.logical(lapply(tax[orders],function(x){x[["tax"]]
[which(x[["pedon"]]==id)]})))) {

    tax[["entisols"]]["tax"][tax[["entisols"]]["pedon"]==id,] <- FALSE

  } else {

    tax[["entisols"]]["tax"][tax[["entisols"]]["pedon"]==id,] <- TRUE

  }

  return(tax)

}
```

Appendix J:

suborders.R

```
# This script contains the functions for classifying taxonomy at the suborder
level.


## NOTE: aquic suborders simplified to having aquic conditions for the time
being



## TODO OVERALL:
# - add check on reaction to alpha,alpha-dipryidyl (variable added later)
# - need to code for: vitrands, torrerts, usters, wassents, and fluvents
# - need to add in the more complex conditions for many aquic suborders
# - check interpretation for cumulative saturation; any hz?
# - need to add conditions for sulfidic materials

# - for taxonomy functions, diag needs to be diags and filtered like hzs







# TODO:
# - interpreted "directly below" as immediately adjacent
gelisols.suborders.histels <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  ## cond1 and cond2
  if(any(hz$cindfragpum) & any(hz$desgnmaster == "O") &
     any(diag$densic$hz,diag$densic$lithic,diag$paralithic$hz)) {

    diag_tops <-
min(diag_top_depth(c("densic","lithic","paralithic"),diag,id))

    cond1 <-  hz$hzdepb[hz$desgnmaster == "O"] >= hz$hzdept[hz$cindfragpum] &
              hz$hzdepb[hz$cindfragpum] == diag_tops

  } else if(any(hz$cindfragpum) & any(hz$desgnmaster == "O") &
            length(hz$hzdepb[hz$desgnmaster == "O"] >=
hz$hzdept[hz$cindfragpum])!=0) {

    cond1 <- FALSE
    sl <- prof_slice(50,0,hz)
    cond2 <- sum(sl$hzthk[which(hz$desgnmaster=="O" | hz$cindfragpum)]) >= 40


  } else {

    cond1 <- FALSE
```

```
    cond2 <- FALSE

  }


  ## cond3
  cond3_contact_tops <-
diag_top_depth(c("glacic","densic","lithic","paralithic"),diag,id)

  if(is.na(cond3_contact_tops)) {

    shallower <- 50

  } else {

    shallower <- min(cond3_contact_tops,na.rm=TRUE)

  }

  sl2 <- prof_slice(shallower,0,hz)
  cond3 <- sum(sl2$hzthk[sl2$desgnmaster == "O"])/sum(sl2$hzthk) >= 0.8



  tax[["histels"]]["cond1"][tax[["histels"]]["pedon"]==id,] <- cond1
  tax[["histels"]]["cond2"][tax[["histels"]]["pedon"]==id,] <- cond2
  tax[["histels"]]["cond3"][tax[["histels"]]["pedon"]==id,] <- cond3


  if(any(cond1,cond2,cond3)) {

    tax[["histels"]]["tax"][tax[["histels"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["histels"]]["tax"][tax[["histels"]]["pedon"]==id,] <- FALSE

  }


  return(tax)




}

gelisols.suborders.turbels <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  cond <- any(hz$cryoturb)
```

```
  if(cond) {

    tax[["turbels"]]["tax"][tax[["turbels"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["turbels"]]["tax"][tax[["turbels"]]["pedon"]==id,] <- FALSE

  }

  return(tax)

}

gelisols.suborders.orthels <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  suborders <- c("histels","turbels")

  if(any(as.logical(lapply(tax[suborders],function(x){x[["tax"]]
[which(x[["pedon"]]==id)]})))) {

    tax[["orthels"]]["tax"][tax[["orthels"]]["pedon"]==id,] <- FALSE

  } else {

    tax[["orthels"]]["tax"][tax[["orthels"]]["pedon"]==id,] <- TRUE

  }

  return(tax)

}




##############################################################################
histosols.suborders.folists <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  cond <- all(hz$cumusat < 30)

  if(cond) {

    tax[["folists"]]["tax"][tax[["folists"]]["pedon"]==id,] <- TRUE

  } else {
```

```
    tax[["folists"]]["tax"][tax[["folists"]]["pedon"]==id,] <- FALSE

  }


  return(tax)


}

# TODO:
# - SKIPPED
# - how to check water potential?
histosols.suborders.wassists <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  return(tax)

}

histosols.suborders.fibrists <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})


  ## cond1 (a OR b) AND cond2 AND COND3

  subsurf_tier_bot <- diag$org_thk$subsurface_tier_bottom
  subsurf_tier_top <- diag$org_thk$subsurface_tier_top

  if(!any(hz$desgnmaster != "O" & hz$hzthk >= 40 &
          (hz$hzdept >= subsurf_tier_top & hz$hzdept <= subsurf_tier_bot)) ) {


    ## cond1a
    subsurf_sl <- prof_slice(subsurf_tier_bot,subsurf_tier_top,hz)
    subsurf_fib_thk <- sum(subsurf_sl$hzthk[subsurf_sl$hzname=="Oi"])
    subsurf_hem_thk <- sum(subsurf_sl$hzthk[subsurf_sl$hzname=="Oe"])
    subsurf_sap_thk <- sum(subsurf_sl$hzthk[subsurf_sl$hzname=="Oa"])

    cond1a <- (subsurf_fib_thk > subsurf_hem_thk & subsurf_fib_thk >
subsurf_sap_thk)
    cond1b <- FALSE


  } else {

    cond1a <- FALSE
    surf_sub_sl <- prof_slice(subsurf_tier_bot,subsurf_tier_top,hz)
    surf_sub_fib_thk <- sum(surf_sub_sl$hzthk[surf_sub_sl$hzname=="Oi"])
    surf_sub_hem_thk <- sum(surf_sub_sl$hzthk[surf_sub_sl$hzname=="Oe"])
```

```
    surf_sub_sap_thk <- sum(surf_sub_sl$hzthk[surf_sub_sl$hzname=="Oa"])

    cond1b <- (surf_sub_fib_thk > surf_sub_hem_thk & surf_sub_fib_thk >
surf_sub_sap_thk)


  }


  tax[["fibrists"]]["cond1a"][tax[["fibrists"]]["pedon"]==id,] <- cond1a
  tax[["fibrists"]]["cond1b"][tax[["fibrists"]]["pedon"]==id,] <- cond1b


  if(any(cond1a,cond1b)) {

    cond1 <- TRUE
    tax[["fibrists"]]["tax"][tax[["fibrists"]]["pedon"]==id,] <- TRUE

  } else {

    cond1 <- FALSE
    tax[["fibrists"]]["tax"][tax[["fibrists"]]["pedon"]==id,] <- FALSE
    tax[["fibrists"]]["tax"][tax[["fibrists"]]["pedon"]==id,] <- FALSE

    return(tax)

  }



  ## cond2

  if(diag$sulfuric$hz) {

    cond2 <- diag$sulfuric$top_depth >= 50

  } else {

    cond2 <- TRUE

  }

  tax[["fibrists"]]["cond2"][tax[["fibrists"]]["pedon"]==id,] <- cond2

  ## cond3 SKIPPED

  if(all(cond1,cond2)) {

    tax[["fibrists"]]["tax"][tax[["fibrists"]]["pedon"]==id,] <- TRUE

  }


  return(tax)
```

```
}

histosols.suborders.saprists <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  subsurf_tier_bot <- diag$org_thk$subsurface_tier_bottom
  subsurf_tier_top <- diag$org_thk$subsurface_tier_top

  if(!any(hz$desgnmaster != "O" & hz$hzthk >= 40 &
          (hz$hzdept >= subsurf_tier_top & hz$hzdept <= subsurf_tier_bot)) ) {


    ## cond1
    subsurf_sl <- prof_slice(subsurf_tier_bot,subsurf_tier_top,hz)
    subsurf_fib_thk <-
sum(subsurf_sl$hzthk[grepl(subsurf_sl$hzname,pattern="Oi")])
    subsurf_hem_thk <-
sum(subsurf_sl$hzthk[grepl(subsurf_sl$hzname,pattern="Oe")])
    subsurf_sap_thk <-
sum(subsurf_sl$hzthk[grepl(subsurf_sl$hzname,pattern="Oa")])


    cond1 <- (subsurf_sap_thk > subsurf_fib_thk & subsurf_sap_thk >
subsurf_hem_thk)
    cond2 <- FALSE


  } else {

    cond1 <- FALSE
    surf_sub_sl <- prof_slice(subsurf_tier_bot,subsurf_tier_top,hz)
    surf_sub_fib_thk <-
sum(surf_sub_sl$hzthk[grepl(surf_sub_sl$hzname,pattern="Oi")])
    surf_sub_hem_thk <-
sum(surf_sub_sl$hzthk[grepl(surf_sub_sl$hzname,pattern="Oe")])
    surf_sub_sap_thk <-
sum(surf_sub_sl$hzthk[grepl(surf_sub_sl$hzname,pattern="Oa")])

    cond2 <- (surf_sub_sap_thk > surf_sub_fib_thk & surf_sub_sap_thk >
surf_sub_hem_thk)


  }


  tax[["saprists"]]["cond1"][tax[["saprists"]]["pedon"]==id,] <- cond1
  tax[["saprists"]]["cond2"][tax[["saprists"]]["pedon"]==id,] <- cond2

  if(any(cond1,cond2)) {
```

```
    tax[["saprists"]]["tax"][tax[["saprists"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["saprists"]]["tax"][tax[["saprists"]]["pedon"]==id,] <- FALSE

  }


  return(tax)



}

histosols.suborders.hemists <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})


  suborders <- c("folists","wassists","fibrists","saprists")

  if(any(as.logical(lapply(tax[suborders],function(x){x[["tax"]]
[which(x[["pedon"]]==id)]})))) {

    tax[["hemists"]]["tax"][tax[["hemists"]]["pedon"]==id,] <- FALSE

  } else {

    tax[["hemists"]]["tax"][tax[["hemists"]]["pedon"]==id,] <- TRUE

  }

  return(tax)

}




################################################################################
spodosols.suborders.aquods <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  minsurf <- diag$orgmin$min_top

  sl1 <- prof_slice(minsurf+50,minsurf,hz)

  if(diag$smr$smr == "aquic"){
```

```
  cond0 <- TRUE
} else {
  cond0 <- any(sl1$aquic)
}

if(!cond0) {

  tax[["aquods"]]["tax"][tax[["aquods"]]["pedon"]==id,] <- FALSE

  return(tax)

}

tax[["aquods"]]["cond0"][tax[["aquods"]]["pedon"]==id,] <- cond0

cond1 <- diag$histic$hz

if(diag$albic$hz & diag$albic$top_depth <= minsurf+50) {

  sl2 <- prof_slice(diag$albic$bottom_depth,diag$albic$top_depth,hz)
  cond2 <- any(sl2$redoxconcen | sl2$redoxdepl)


} else if(diag$spodic$hz & diag$spodic$top_depth <= minsurf+50) {

  sl2 <- prof_slice(diag$spodic$bottom_depth,diag$spodic$top_depth,hz)
  cond2 <- any(sl2$redoxconcen | sl2$redoxdepl)


} else {

  cond2 <- FALSE

}

tax[["aquods"]]["cond1"][tax[["aquods"]]["pedon"]==id,] <- cond1
tax[["aquods"]]["cond2"][tax[["aquods"]]["pedon"]==id,] <- cond2


if(any(cond1,cond2)) {

  tax[["aquods"]]["tax"][tax[["aquods"]]["pedon"]==id,] <- TRUE

} else {

  tax[["aquods"]]["tax"][tax[["aquods"]]["pedon"]==id,] <- FALSE

}


return(tax)
```

```
}

spodosols.suborders.gelods <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  cond <- diag$str$str == "gelic"

  if(cond) {

    tax[["gelods"]]["tax"][tax[["gelods"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["gelods"]]["tax"][tax[["gelods"]]["pedon"]==id,] <- FALSE

  }


  return(tax)


}

spodosols.suborders.cryods <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  cond <- diag$str$str == "cryic"

  if(cond) {

    tax[["cryods"]]["tax"][tax[["cryods"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["cryods"]]["tax"][tax[["cryods"]]["pedon"]==id,] <- FALSE

  }


  return(tax)

}

spodosols.suborders.humods <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  sl <- prof_slice(diag$spodic$bottom_depth,diag$spodic$top_depth,hz)
  cond <- any(sl$oc >= 6 & sl$hzthk >= 10)
```

```
  if(cond) {

    tax[["humods"]]["tax"][tax[["humods"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["humods"]]["tax"][tax[["humods"]]["pedon"]==id,] <- FALSE

  }


  return(tax)


}

spodosols.suborders.orthods <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  suborders <- c("aquods","gelods","cryods","humods","orthods")

  if(any(as.logical(lapply(tax[suborders],function(x){x[["tax"]]
[which(x[["pedon"]]==id)]})))) {

    tax[["orthods"]]["tax"][tax[["orthods"]]["pedon"]==id,] <- FALSE

  } else {

    tax[["orthods"]]["tax"][tax[["orthods"]]["pedon"]==id,] <- TRUE

  }

  return(tax)

}




################################################################################
# TODO:
# - simplified to aquic conditions at certain depth
# - need to include more complex conditions
andisols.suborders.aquands <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  sl <- prof_slice(50,0,hz)
```

```
  cond <- any(sl$aquic)

  if(cond) {

    tax[["aquands"]]["tax"][tax[["aquands"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["aquands"]]["tax"][tax[["aquands"]]["pedon"]==id,] <- FALSE

  }

  return(tax)

}

andisols.suborders.gelands <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  cond <- diag$str$str == "gelic"

  if(cond) {

    tax[["gelands"]]["tax"][tax[["gelands"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["gelands"]]["tax"][tax[["gelands"]]["pedon"]==id,] <- FALSE

  }


  return(tax)

}

andisols.suborders.cryands <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  cond <- diag$str$str == "cryic"

  if(cond) {

    tax[["cryands"]]["tax"][tax[["cryands"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["cryands"]]["tax"][tax[["cryands"]]["pedon"]==id,] <- FALSE

  }
```

```
  return(tax)

}

andisols.suborders.torrands <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  cond <- diag$smr$smr == "aridic"

  if(cond) {

    tax[["torrands"]]["tax"][tax[["torrands"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["torrands"]]["tax"][tax[["torrands"]]["pedon"]==id,] <- FALSE

  }


  return(tax)

}

andisols.suborders.xerands <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  cond <- diag$smr$smr == "xeric"

  if(cond) {

    tax[["xerands"]]["tax"][tax[["xerands"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["xerands"]]["tax"][tax[["xerands"]]["pedon"]==id,] <- FALSE

  }


  return(tax)

}

# TODO:
# - SKIPPED
andisols.suborders.vitrands <- function(hzs, diags, tax, id) {
```

```
  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  return(tax)

}

andisols.suborders.ustands <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  cond <- diag$smr$smr == "ustic"

  if(cond) {

    tax[["ustands"]]["tax"][tax[["ustands"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["ustands"]]["tax"][tax[["ustands"]]["pedon"]==id,] <- FALSE

  }


  return(tax)

}

andisols.suborders.udands <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  suborders <-
c("aquands","gelands","cryands","torrands","xerands","vitrands","ustands","ustands")

  if(any(as.logical(lapply(tax[suborders],function(x){x[["tax"]]
[which(x[["pedon"]]==id)]})))) {

    tax[["udands"]]["tax"][tax[["udands"]]["pedon"]==id,] <- FALSE

  } else {

    tax[["udands"]]["tax"][tax[["udands"]]["pedon"]==id,] <- TRUE

  }

  return(tax)

}
```

```
#############################################################################
# TODO:
# - simplified to aquic conditions at certain depth
# - need to include more complex conditions
oxisols.suborders.aquox <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  minsurf <- diag$orgmin$min_top

  sl <- prof_slice(minsurf+50,minsurf,hz)

  cond <- any(sl$aquic)

  if(cond) {

    tax[["aquox"]]["tax"][tax[["aquox"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["aquox"]]["tax"][tax[["aquox"]]["pedon"]==id,] <- FALSE

  }

  return(tax)

}

oxisols.suborders.torrox <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  cond <- diag$smr$smr == "aridic"

  if(cond) {

    tax[["torrox"]]["tax"][tax[["torrox"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["torrox"]]["tax"][tax[["torrox"]]["pedon"]==id,] <- FALSE

  }


  return(tax)

}

oxisols.suborders.ustox <- function(hzs, diags, tax, id) {
```

```
  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  cond <- diag$smr$smr %in% c("ustic","xeric")

  if(cond) {

    tax[["ustox"]]["tax"][tax[["ustox"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["ustox"]]["tax"][tax[["ustox"]]["pedon"]==id,] <- FALSE

  }


  return(tax)

}

oxisols.suborders.perox <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  cond <- diag$smr$smr == "perudic"

  if(cond) {

    tax[["perox"]]["tax"][tax[["perox"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["perox"]]["tax"][tax[["perox"]]["pedon"]==id,] <- FALSE

  }


  return(tax)

}

oxisols.suborders.udox <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  suborders <- c("aquox","torrox","ustox","perox")

  if(any(as.logical(lapply(tax[suborders],function(x){x[["tax"]]
[which(x[["pedon"]]==id)]})))) {

    tax[["udox"]]["tax"][tax[["udox"]]["pedon"]==id,] <- FALSE
```

```
  } else {

    tax[["udox"]]["tax"][tax[["udox"]]["pedon"]==id,] <- TRUE

  }

  return(tax)

}




################################################################################
vertisols.suborders.aquerts <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  minsurf <- diag$orgmin$min_top

  sl <- prof_slice(minsurf+50,minsurf,hz)

  cond0 <- any(sl$aquic)


  cond1 <- ((hz$colorchroma_d <= 2 | hz$colorchroma_w <= 2) & hz$redoxconcen)
|
           ((hz$colorchroma_d <= 1 | hz$colorchroma_w <= 1))


  if(sum(hz$hzthk[cond1])) {

    cond1 <- TRUE

  } else {

    cond1 <- FALSE

  }


  tax[["aquerts"]]["cond0"][tax[["aquerts"]]["pedon"]==id,] <- cond0
  tax[["aquerts"]]["cond1"][tax[["aquerts"]]["pedon"]==id,] <- cond1


  if(cond0 & cond1) {

    tax[["aquerts"]]["tax"][tax[["aquerts"]]["pedon"]==id,] <- TRUE

  } else {
```

```
    tax[["aquerts"]]["tax"][tax[["aquerts"]]["pedon"]==id,] <- FALSE

  }

  return(tax)

}

vertisols.suborders.cryerts <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  cond <- diag$str$str == "cryic"

  if(cond) {

    tax[["cryerts"]]["tax"][tax[["cryerts"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["cryerts"]]["tax"][tax[["cryerts"]]["pedon"]==id,] <- FALSE

  }


  return(tax)



}

# TODO:
# - cond2 skipped: how to check for cracks?
# - shortcut with SMR
vertisols.suborders.xererts <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  cond <- diag$str$str %in% c("thermic","mesic","frigid") & diag$smr$smr ==
"xeric"

  if(cond) {

    tax[["xererts"]]["tax"][tax[["xererts"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["xererts"]]["tax"][tax[["xererts"]]["pedon"]==id,] <- FALSE

  }
```

```
  return(tax)

}

# TODO:
# - SKIPPED
vertisols.suborders.torrerts <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  return(tax)

}

# TODO:
# - shortcut with sMR
vertisols.suborders.usterts <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  cond <- diag$smr$smr == "ustic"

  if(cond) {

    tax[["usterts"]]["tax"][tax[["usterts"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["usterts"]]["tax"][tax[["usterts"]]["pedon"]==id,] <- FALSE

  }

  return(tax)

}

vertisols.suborders.uderts <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  suborders <- c("aquerts","cryerts","xererts","torrerts","usterts")

  if(any(as.logical(lapply(tax[suborders],function(x){x[["tax"]]
[which(x[["pedon"]]==id)]})))) {

    tax[["uderts"]]["tax"][tax[["uderts"]]["pedon"]==id,] <- FALSE

  } else {

    tax[["uderts"]]["tax"][tax[["uderts"]]["pedon"]==id,] <- TRUE
```

```
  }

  return(tax)

}




################################################################################
aridisols.suborders.cryids <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  cond <- diag$str$str == "cryic"

  if(cond) {

    tax[["cryids"]]["tax"][tax[["cryids"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["cryids"]]["tax"][tax[["cryids"]]["pedon"]==id,] <- FALSE

  }


  return(tax)

}

aridisols.suborders.salids <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  minsurf <- diag$orgmin$min_top

  if(diag$salic$hz) {

    cond <- diag$salic$top_depth <= minsurf+100


  } else {

    cond <- FALSE

  }

  tax[["salids"]]["tax"][tax[["salids"]]["pedon"]==id,] <- cond

  return(tax)
```

```
}

aridisols.suborders.durids <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  minsurf <- diag$orgmin$min_top

  if(diag$duripan$hz) {

    cond <- diag$duripan$top_depth <= minsurf+100


  } else {

    cond <- FALSE

  }

  tax[["durids"]]["tax"][tax[["durids"]]["pedon"]==id,] <- cond


  return(tax)

}

aridisols.suborders.gypsids <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  minsurf <- diag$orgmin$min_top

  if(diag$gypsic$hz | diag$petrogypsic$hz) {

    cond <- (diag$gypsic$top_depth <= minsurf+100 | diag$petrogypsic$top_depth
<= minsurf+100) &
      (diag$petrocalcic$top_depth <= diag$gypsic$top_depth |
       diag$petrocalcic$top_depth <= diag$petrogypsic$top_depth)


  } else {

    cond <- FALSE

  }

  tax[["gypsids"]]["tax"][tax[["gypsids"]]["pedon"]==id,] <- cond


  return(tax)
```

```
}

aridisols.suborders.argids <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  minsurf <- diag$orgmin$min_top

  if(!diag$petrocalcic$hz) {

    cond <- any(diag$argillic$hz,diag$natric$hz)

  } else {

    cond <- any(diag$argillic$hz,diag$natric$hz) & !
(diag$petrocalcic$top_depth <= minsurf+100)

  }

  tax[["argids"]]["tax"][tax[["argids"]]["pedon"]==id,] <- cond

  return(tax)

}

aridisols.suborders.calcids <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  minsurf <- diag$orgmin$min_top

  if(diag$calcic$hz | diag$petrocalcic$hz) {

    cond <- diag$calcic$top_depth <= minsurf+100 | diag$petrocalcic$top_depth
<= minsurf+100


  } else {

    cond <- FALSE

  }

  tax[["calcids"]]["tax"][tax[["calcids"]]["pedon"]==id,] <- cond

  return(tax)

}

aridisols.suborders.cambids <- function(hzs, diags, tax, id) {
```

```
  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  suborders <- c("cryids","salids","durids","gypsids","argids","calcids")

  if(any(as.logical(lapply(tax[suborders],function(x){x[["tax"]]
[which(x[["pedon"]]==id)]})))) {

    tax[["cambids"]]["tax"][tax[["cambids"]]["pedon"]==id,] <- FALSE

  } else {

    tax[["cambids"]]["tax"][tax[["cambids"]]["pedon"]==id,] <- TRUE

  }

  return(tax)


}




################################################################################
# TODO:
# - simplified to aquic conditions at certain depth
# - need to include more complex conditions
ultisols.suborders.aquults <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  minsurf <- diag$orgmin$min_top

  sl <- prof_slice(minsurf+40,25,hz)

  cond <- any(sl$aquic)

  if(cond) {

    tax[["aquults"]]["tax"][tax[["aquults"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["aquults"]]["tax"][tax[["aquults"]]["pedon"]==id,] <- FALSE

  }

  return(tax)

}
```

```
ultisols.suborders.humults <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  minsurf <- diag$orgmin$min_top

  if(diag$argillic$hz) {

    sl <- prof_slice(diag$argillic$top_depth+15,diag$argillic$top_depth,hz)
    cond1 <- weighted.mean(sl$oc,sl$hzthk/sum(sl$hzthk)) >= 0.9

  } else if(diag$kandic$hz) {

    sl <- prof_slice(diag$kandicc$top_depth+15,diag$kandic$top_depth,hz)
    cond1 <- weighted.mean(sl$oc,sl$hzthk/sum(sl$hzthk)) >= 0.9

  }

  cond2_sl <- prof_slice(minsurf+100,minsurf,hz)
  oc_t <- (cond2_sl$dbovendry * (cond2_sl$oc/100)) * cond2_sl$hzthk * 10 # kg/
m2
  cond2 <- sum(oc_t,na.rm=TRUE) >= 12 # removing NA could produce inaccurate
results


  if(any(cond1,cond2)) {

    tax[["humults"]]["tax"][tax[["humults"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["humults"]]["tax"][tax[["humults"]]["pedon"]==id,] <- FALSE

  }

  return(tax)

}

ultisols.suborders.udults <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  cond <- diag$smr$smr == "udic"

  if(cond) {

    tax[["udults"]]["tax"][tax[["udults"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["udults"]]["tax"][tax[["udults"]]["pedon"]==id,] <- FALSE
```

```
  }

  return(tax)

}

ultisols.suborders.ustults <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  cond <- diag$smr$smr == "ustic"

  if(cond) {

    tax[["ustults"]]["tax"][tax[["ustults"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["ustults"]]["tax"][tax[["ustults"]]["pedon"]==id,] <- FALSE

  }


  return(tax)

}

ultisols.suborders.xerults <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  suborders <- c("aquults","humults","udults","ustults")

  if(any(as.logical(lapply(tax[suborders],function(x){x[["tax"]]
[which(x[["pedon"]]==id)]})))) {

    tax[["xerults"]]["tax"][tax[["xerults"]]["pedon"]==id,] <- FALSE

  } else {

    tax[["xerults"]]["tax"][tax[["xerults"]]["pedon"]==id,] <- TRUE

  }

  return(tax)

}
```

```
#############################################################################
# TODO:
# - dry or moist chroma? checks both
# - skipped last condition of cond1; separates hz that would otherwise be
mollic
# - aquic flag used; appropriate here?
mollisols.suborders.albolls <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})


  minsurf <- diag$orgmin$min_top


  cond1 <- diag$argillic$hz | diag$natric$hz

  if(!cond1) {

    tax[["albolls"]]["cond1"][tax[["albolls"]]["pedon"]==id,] <- cond1
    tax[["albolls"]]["tax"][tax[["albolls"]]["pedon"]==id,] <- FALSE

    return(tax)

  }



  ## cond2

  if(diag$albic$hz) {

    albic <- prof_slice(diag$albic$bottom_depth,diag$albic$top_depth,hz)
    cond2 <- all((albic$colorchroma_d <= 2 | albic$colorchroma_w <= 2)) &
             diag$albic$thick >= 2.5 &
             diag$albic$bottom_depth >= minsurf+18
             # & diag$albic$top_depth == diag$mollic$bottom_depth


  } else {

    cond2 <- FALSE

  }


  if(!cond2) {

    tax[["albolls"]]["cond2"][tax[["albolls"]]["pedon"]==id,] <- cond2
    tax[["albolls"]]["tax"][tax[["albolls"]]["pedon"]==id,] <- FALSE
```

```
    return(tax)

  }


  ## cond3


  argnat_top <- min(diag_top_depth(c("argillic","natric"),diag,id),na.rm=TRUE)
  argnat_bot <- min(diag_bot_depth(c("argillic","natric"),diag,id),na.rm=TRUE)
  argnat <- prof_slice(argnat_bot,argnat_top,hz)

  cond3 <- any((albic$redoxconcen & albic$hzdept <= minsurf+100 & albic$aquic)
|
           (argnat$redoxconcen & argnat$hzdept <= minsurf+100 & argnat$aquic))



  ## cond4

  cond4 <- !diag$str$str %in% c("gelic","cryic")



  tax[["albolls"]]["cond3"][tax[["albolls"]]["pedon"]==id,] <- cond3
  tax[["albolls"]]["cond4"][tax[["albolls"]]["pedon"]==id,] <- cond4

  if(all(cond1,cond2,cond3,cond4)) {

    tax[["albolls"]]["tax"][tax[["albolls"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["albolls"]]["tax"][tax[["albolls"]]["pedon"]==id,] <- FALSE

  }


  return(tax)




}

# TODO:
# - assumed dry or moist colors if not stated
# - check 5a2 and 5b2 color moisture state wording
mollisols.suborders.aquolls <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})
```

```
minsurf <- diag$orgmin$min_top

## cond0

contacts <- min(diag_top_depth(c("densic","lithic","paralithic"),diag,id))

if(is.na(contacts)) {

  shallower_top <- minsurf+40
  shallower_bot <- minsurf+50

} else if(contacts < minsurf+50) {

  shallower_top <- 0
  shallower_bot <- contacts

} else {

  shallower_top <- minsurf+40
  shallower_bot <- minsurf+50

}

sl <- prof_slice(shallower_bot,shallower_top,hz)

if(diag$smr$smr == "aquic"){
  cond0 <- TRUE
} else {
  cond0 <- any(sl$aquic)
}



### If mollic epi is entire profile, check smr as shortcut ###
if(diag$mollic$bottom_depth == max(hz$hzdepb)) {
  tax[["aquolls"]]["cond0"][tax[["aquolls"]]["pedon"]==id,] <- cond0

  if(diag$smr$smr == "aquic") {
    tax[["aquolls"]]["tax"][tax[["aquolls"]]["pedon"]==id,] <- TRUE
  } else {
    tax[["aquolls"]]["tax"][tax[["aquolls"]]["pedon"]==id,] <- FALSE
  }

  return(tax)
}



## cond1

if(diag$histic$hz) {

  cond1 <- diag$histic$top_depth >= diag$mollic$top_depth
```

```
  } else {

    cond1 <- FALSE

  }


  ## cond2

  mollic_1half <-
prof_slice(diag$mollic$top_depth+(0.5*diag$mollic$thick),diag$mollic$top_depth,hz)
# top half of mollic

  cond2_1 <- any((mollic_1half$esp >= 15 | mollic_1half$sar >= 13))

  sl1 <- prof_slice(minsurf+50,minsurf,hz)
  sl2 <- prof_slice(max(hz$hzdepb),minsurf+50,hz)

  cond2_2 <- sl1$esp[nrow(sl1)] > sl2$esp[1] | sl1$sar[nrow(sl1)] > sl2$sar[1]


  if(cond2_1 & cond2_2) {
    cond2 <- TRUE
  } else {
    cond2 <- FALSE
  }


  ## cond3

  if(diag$calcic$hz | diag$petrocalcic$hz) {

    cond3 <- any(diag$calcic$top_depth <= minsurf+40 |
diag$petrocalcic$top_depth <= minsurf+40,na.rm=TRUE)

  } else {

    cond3 <- FALSE

  }



  ## cond4

  if(diag$lithic$hz) {

    mollic <- prof_slice(diag$mollic$bottom_depth,diag$mollic$top_depth,hz)
    cond4 <- all((mollic$colorchroma_d <= 1 | mollic$colorchroma_w <= 1) &
diag$lithic$top_depth <= minsurf+30 &
             diag$mollic$bottom_depth == diag$lithic$top_depth)

  } else {
```

```
    cond4 <- FALSE

  }



  ## cond5

  # a (a1 OR a2 (a2a OR a2b OR a2c OR a2d OR a2e OR a2f OR a2g)) OR b (b1 OR
b2 (b2a OR b2b OR b2c))

  mollic_2half <-
prof_slice(diag$mollic$bottom_depth,diag$mollic$top_depth+(0.5*diag$mollic$thick),hz)
# bottom half of mollic

  ## cond5a

  cond5a <- any(mollic_2half$colorchroma_d <= 1 | mollic_2half$colorchroma_w
<= 1)

  cond5a1 <- any(mollic_2half$redoxconcen)

  if(diag$calcic$hz) {

    cond5a2_sl <- prof_slice(minsurf+75,minsurf,hz)

  } else {

    cond5a2_sl <- hz[min(which(hz$hzdepb > diag$mollic$bottom_depth &
hz$hzdept >= diag$mollic$bottom_depth)),] # single hz

  }

  cond5a2 <- any(cond5a2_sl$colorvalue_w >= 4)

  cond5a2a <- all((cond5a2_sl$colorchroma_d == 1 | cond5a2_sl$colorchroma_w ==
1) &
                  (cond5a2_sl$colorhue_d %in%
c("10YR","7.5YR","5YR","10R","7.5R","5R","2.5R") |
                   cond5a2_sl$colorhue_w %in%
c("10YR","7.5YR","5YR","10R","7.5R","5R","2.5R")) &
                   (cond5a2_sl$redoxconcen))

  cond5a2b <- all((cond5a2_sl$colorchroma_d <= 2 | cond5a2_sl$colorchroma_w <=
2) &
                    (cond5a2_sl$colorhue_d == "2.5Y" |
                       cond5a2_sl$colorhue_w == "2.5Y") &
                    (cond5a2_sl$redoxconcen))

  cond5a2c <- all((cond5a2_sl$colorchroma_d == 1 | cond5a2_sl$colorchroma_w ==
1) &
                  (cond5a2_sl$colorhue_d %in% c("2.5Y","5Y","7.5Y","10Y") |
                   cond5a2_sl$colorhue_w %in% c("2.5Y","5Y","7.5Y","10Y")))
```

```
  cond5a2d <- all((cond5a2_sl$colorchroma_d <= 3 | cond5a2_sl$colorchroma_w <=
3) &
                  (cond5a2_sl$colorhue_d == "5Y" |
                   cond5a2_sl$colorhue_w == "5Y") &
                   (cond5a2_sl$redoxconcen))

  cond5a2e <- all((cond5a2_sl$colorchroma_d == 0 | cond5a2_sl$colorchroma_w ==
0) &
                    (cond5a2_sl$colorhue_d == "N" |
                     cond5a2_sl$colorhue_w == "N"))

  cond5a2f <- all((cond5a2_sl$colorhue_d %in% c("5GY","5G","5BG","5B") |
                   cond5a2_sl$colorhue_w %in% c("5GY","5G","5BG","5B")))

  ## cond5a2g not checked


  if(cond5a & cond5a2 &
any(cond5a2a,cond5a2b,cond5a2c,cond5a2d,cond5a2e,cond5a2f)) {
    cond5a2 <- TRUE
  } else {
    cond5a2 <- FALSE
  }

  if(cond5a & any(cond5a1,cond5a2)) {
    cond5a <- TRUE
  } else {
    cond5a <- FALSE
  }



  ## cond5b

  cond5b <- any(mollic_2half$colorchroma_d == 2 | mollic_2half$colorchroma_w
== 2)

  cond5b1 <- any(mollic_2half$redoxconcen)


  cond5b2_sl <- hz[min(which(hz$hzdepb > diag$mollic$bottom_depth & hz$hzdept
>= diag$mollic$bottom_depth)),] # single hz

  cond5b2a <- all(cond5b2_sl$colorvalue_w == 4 &
                (cond5b2_sl$colorchroma_d == 2 | cond5b2_sl$colorchroma_w ==
2) &
                (cond5b2_sl$redoxdepl & cond5b2_sl$redoxcolorvalue >= 4 &
cond5b2_sl$redoxcolorchroma <= 1))

  cond5b2b <- all(cond5b2_sl$colorvalue_w >= 5 &
                 (cond5b2_sl$colorchroma_d <= 2 | cond5b2_sl$colorchroma_w <=
2) &
                  cond5b2_sl$redoxconcen)
```

```
  cond5b2c <- all(cond5b2_sl$colorvalue_w == 4 &
                  (cond5b2_sl$colorchroma_d <= 1 | cond5b2_sl$colorchroma_w <=
1))

  if(cond5b & any(cond5b2a,cond5b2b,cond5b2c)) {
    cond5b2 <- TRUE
  } else {
    cond5b2 <- FALSE
  }

  if(cond5b & any(cond5b1,cond5b2)) {
    cond5b <- TRUE
  } else {
    cond5b <- FALSE
  }


  if(any(cond5a,cond5b)) {
    cond5 <- TRUE
  } else {
    cond5 <- FALSE
  }



  ## cond6
  cond6 <- any(sl$aadipyridyl)


  tax[["aquolls"]]["cond0"][tax[["aquolls"]]["pedon"]==id,] <- cond0
  tax[["aquolls"]]["cond1"][tax[["aquolls"]]["pedon"]==id,] <- cond1
  tax[["aquolls"]]["cond2"][tax[["aquolls"]]["pedon"]==id,] <- cond2
  tax[["aquolls"]]["cond3"][tax[["aquolls"]]["pedon"]==id,] <- cond3
  tax[["aquolls"]]["cond4"][tax[["aquolls"]]["pedon"]==id,] <- cond4
  tax[["aquolls"]]["cond5"][tax[["aquolls"]]["pedon"]==id,] <- cond5
  tax[["aquolls"]]["cond5a"][tax[["aquolls"]]["pedon"]==id,] <- cond5a
  tax[["aquolls"]]["cond5a1"][tax[["aquolls"]]["pedon"]==id,] <- cond5a1
  tax[["aquolls"]]["cond5a2"][tax[["aquolls"]]["pedon"]==id,] <- cond5a2
  tax[["aquolls"]]["cond5a2a"][tax[["aquolls"]]["pedon"]==id,] <- cond5a2a
  tax[["aquolls"]]["cond5a2b"][tax[["aquolls"]]["pedon"]==id,] <- cond5a2b
  tax[["aquolls"]]["cond5a2c"][tax[["aquolls"]]["pedon"]==id,] <- cond5a2c
  tax[["aquolls"]]["cond5a2d"][tax[["aquolls"]]["pedon"]==id,] <- cond5a2d
  tax[["aquolls"]]["cond5a2e"][tax[["aquolls"]]["pedon"]==id,] <- cond5a2e
  tax[["aquolls"]]["cond5a2f"][tax[["aquolls"]]["pedon"]==id,] <- cond5a2f
  tax[["aquolls"]]["cond5b"][tax[["aquolls"]]["pedon"]==id,] <- cond5b
  tax[["aquolls"]]["cond5b1"][tax[["aquolls"]]["pedon"]==id,] <- cond5b1
  tax[["aquolls"]]["cond5b2"][tax[["aquolls"]]["pedon"]==id,] <- cond5b2
  tax[["aquolls"]]["cond5b2a"][tax[["aquolls"]]["pedon"]==id,] <- cond5b2a
  tax[["aquolls"]]["cond5b2b"][tax[["aquolls"]]["pedon"]==id,] <- cond5b2b
  tax[["aquolls"]]["cond5b2c"][tax[["aquolls"]]["pedon"]==id,] <- cond5b2c
  tax[["aquolls"]]["cond6"][tax[["aquolls"]]["pedon"]==id,] <- cond6

  if(cond0 & any(cond1,cond2,cond3,cond4,cond5,cond6)) {
```

```
    tax[["aquolls"]]["tax"][tax[["aquolls"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["aquolls"]]["tax"][tax[["aquolls"]]["pedon"]==id,] <- FALSE

  }


  return(tax)


}

mollisols.suborders.rendolls <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})


  ## cond1

  cond1 <- diag$mollic$thick <= 50



  ## cond2

  cond2 <- !any(diag$argillic$hz,diag$calcic$hz)


  ## cond3

  hz_bl_mollic <- which.min(hz$hzdepb > diag$mollic$bottom_depth)
  sl <- prof_slice(hz$hzdepb[hz_bl_mollic],diag$mollic$top_depth,hz)

  cond3 <- any(sl$caco3 >= 40)


  ## cond4

  cond4a <- diag$smr$smr == "udic"
  cond4b <- diag$str$str == "cryic"

  if(any(cond4a,cond4b)) {

    cond4 <- TRUE

  } else {

    cond4 <- FALSE

  }
```

```
  tax[["rendolls"]]["cond1"][tax[["rendolls"]]["pedon"]==id,] <- cond1
  tax[["rendolls"]]["cond2"][tax[["rendolls"]]["pedon"]==id,] <- cond2
  tax[["rendolls"]]["cond3"][tax[["rendolls"]]["pedon"]==id,] <- cond3
  tax[["rendolls"]]["cond4"][tax[["rendolls"]]["pedon"]==id,] <- cond4
  tax[["rendolls"]]["cond4a"][tax[["rendolls"]]["pedon"]==id,] <- cond4a
  tax[["rendolls"]]["cond4b"][tax[["rendolls"]]["pedon"]==id,] <- cond4b


  if(all(cond1,cond2,cond3,cond4)) {

    tax[["rendolls"]]["tax"][tax[["rendolls"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["rendolls"]]["tax"][tax[["rendolls"]]["pedon"]==id,] <- FALSE

  }



  return(tax)



}

mollisols.suborders.gelolls <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  cond <- diag$str$str == "gelic"

  if(cond) {

    tax[["gelolls"]]["tax"][tax[["gelolls"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["gelolls"]]["tax"][tax[["gelolls"]]["pedon"]==id,] <- FALSE

  }


  return(tax)

}

mollisols.suborders.cryolls <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})
```

```
  cond <- diag$str$str == "cryic"

  if(cond) {

    tax[["cryolls"]]["tax"][tax[["cryolls"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["cryolls"]]["tax"][tax[["cryolls"]]["pedon"]==id,] <- FALSE

  }


  return(tax)

}

# TODO:
# - how to check if SMR "borders on xeric"?
mollisols.suborders.xerolls <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  cond <- diag$smr$smr == "xeric"

  if(cond) {

    tax[["xerolls"]]["tax"][tax[["xerolls"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["xerolls"]]["tax"][tax[["xerolls"]]["pedon"]==id,] <- FALSE

  }


  return(tax)

}

# TODO:
# - how to check if SMR "borders on ustic"?
mollisols.suborders.ustolls <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  cond <- diag$smr$smr == "ustic"

  if(cond) {

    tax[["ustolls"]]["tax"][tax[["ustolls"]]["pedon"]==id,] <- TRUE
```

```
  } else {

    tax[["ustolls"]]["tax"][tax[["ustolls"]]["pedon"]==id,] <- FALSE

  }


  return(tax)

}

mollisols.suborders.udolls <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  suborders <-
c("albolls","aquolls","rendolls","gelolls","cryolls","xerolls","ustolls")

  if(any(as.logical(lapply(tax[suborders],function(x){x[["tax"]]
[which(x[["pedon"]]==id)]})))) {

    tax[["udolls"]]["tax"][tax[["udolls"]]["pedon"]==id,] <- FALSE

  } else {

    tax[["udolls"]]["tax"][tax[["udolls"]]["pedon"]==id,] <- TRUE

  }

  return(tax)

}




################################################################################
# TODO:
# - also requiring aquic moisture regime
alfisols.suborders.aqualfs <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  minsurf <- diag$orgmin$min_top

  sl <- prof_slice(minsurf+50,minsurf,hz)

  cond0 <- any(sl$aquic) & diag$smr$smr == "aquic"
```

```
  # 1 (1a OR 1b OR 1c) OR 2

  ## cond1

  cond1_ap <- hz$hzdepb[grep(hz$hzname,pattern="[p]")]

  if(length(cond1_ap)!=0) {

    cond1_deeper <- max(cond1_ap,minsurf+25)

  } else {

    cond1_deeper <- minsurf+25

  }

  cond1_sl <- prof_slice(40,cond1_deeper,hz)
  cond1 <- any(all(cond1_sl$redoxconcen) | all(cond1_sl$redoxconcre) |
                 all(cond1_sl$redoxdepl) | all(cond1_sl$redoxmass) |
                 all(grepl(cond1_sl$hzname,pattern="g")))

  if(cond1) {

    cond1_up_diag <-
diag_top_depth(c("argillic","natric","glossic","kandic"),diag,id)

    if(!is.na(cond1_up_diag)) {


      ## trying this out instead
      for(n in names(cond1_up_diag)) {

        diag_sl <- prof_slice(diag[[n]][['top_depth']]+12.5,diag[[n]]
[['top_depth']],hz)

        cond1a <- any((diag_sl$redoxdepl|grepl(diag_sl$hzname,pattern="g")) &
                        diag_sl$colorchroma_w <= 2,na.rm=TRUE)

        cond1b <- any(diag_sl$redoxconcen & diag_sl$redoxcolorchroma <=
2,na.rm=TRUE)

        ## no vars to differentiate between matrix and ped redox
        cond1c <- any((diag_sl$redoxdepl|grepl(diag_sl$hzname,pattern="g")) &
                        diag_sl$colorchroma_w <= 1,na.rm=TRUE)

        if(any(cond1a,cond1b,cond1c)) {
          break
        }

      }


    } else {
```

```
      cond1 <- FALSE
      cond1a <- FALSE
      cond1b <- FALSE
      cond1c <- FALSE

    }

  } else {

    cond1 <- FALSE
    cond1a <- FALSE
    cond1b <- FALSE
    cond1c <- FALSE

  }



  cond2 <- all(sl$aadipyridyl[sl$aquic])


  tax[["aqualfs"]]["cond0"][tax[["aqualfs"]]["pedon"]==id,] <- cond0
  tax[["aqualfs"]]["cond1"][tax[["aqualfs"]]["pedon"]==id,] <- cond1
  tax[["aqualfs"]]["cond1a"][tax[["aqualfs"]]["pedon"]==id,] <- cond1a
  tax[["aqualfs"]]["cond1b"][tax[["aqualfs"]]["pedon"]==id,] <- cond1b
  tax[["aqualfs"]]["cond1c"][tax[["aqualfs"]]["pedon"]==id,] <- cond1c
  tax[["aqualfs"]]["cond2"][tax[["aqualfs"]]["pedon"]==id,] <- cond2


  if(cond0 & ((cond1 & any(cond1a,cond1b,cond1c)) | cond2)) {

    tax[["aqualfs"]]["tax"][tax[["aqualfs"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["aqualfs"]]["tax"][tax[["aqualfs"]]["pedon"]==id,] <- FALSE

  }


  return(tax)


}

alfisols.suborders.cryalfs <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  cond <- diag$str$str %in% c("cryic","isofrigid")

  if(cond) {
```

```
    tax[["cryalfs"]]["tax"][tax[["cryalfs"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["cryalfs"]]["tax"][tax[["cryalfs"]]["pedon"]==id,] <- FALSE

  }


  return(tax)

}

alfisols.suborders.ustalfs <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  cond <- diag$smr$smr == "ustic"

  if(cond) {

    tax[["ustalfs"]]["tax"][tax[["ustalfs"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["ustalfs"]]["tax"][tax[["ustalfs"]]["pedon"]==id,] <- FALSE

  }


  return(tax)

}

alfisols.suborders.xeralfs <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  cond <- diag$smr$smr == "xeric"

  if(cond) {

    tax[["xeralfs"]]["tax"][tax[["xeralfs"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["xeralfs"]]["tax"][tax[["xeralfs"]]["pedon"]==id,] <- FALSE

  }


  return(tax)
```

```
}

alfisols.suborders.udalfs <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  suborders <- c("aqualfs","cryalfs","ustalfs","xeralfs")

  if(any(as.logical(lapply(tax[suborders],function(x){x[["tax"]]
[which(x[["pedon"]]==id)]})))) {

    tax[["udalfs"]]["tax"][tax[["udalfs"]]["pedon"]==id,] <- FALSE

  } else {

    tax[["udalfs"]]["tax"][tax[["udalfs"]]["pedon"]==id,] <- TRUE

  }

  return(tax)

}




################################################################################
# TODO:
# - assumed dry chroma
# - how to check for groundwater in cond2? skipped for now
inceptisols.suborders.aquepts <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})


  ## cond1 (a OR b OR c OR d) OR cond2


  minsurf <- diag$orgmin$min_top

  ## cond1

  contacts <- min(diag_top_depth(c("densic","lithic","paralithic"),diag,id))

  if(is.na(contacts)) {

    shallower_top <- minsurf+40
    shallower_bot <- minsurf+50

  } else if(contacts < minsurf+50) {
```

```
    shallower_top <- 0
    shallower_bot <- contacts

  } else {

    shallower_top <- minsurf+40
    shallower_bot <- minsurf+50

  }


  if(shallower_top > max(hz$hzdepb)) {

    cond1 <- FALSE
    tax[["aquepts"]]["cond1"][tax[["aquepts"]]["pedon"]==id,] <- cond1
    tax[["aquepts"]]["tax"][tax[["aquepts"]]["pedon"]==id,] <- FALSE

    return(tax)

  }


  sl <- prof_slice(shallower_bot,shallower_top,hz)
  cond1 <- any(sl$aquic)


  ## cond1a

  cond1a <- diag$histic$hz


  ## cond1b

  if(diag$sulfuric$hz) {

    cond1b <- diag$sulfuric$top_depth <= minsurf+50

  } else {

    cond1b <- FALSE

  }


  ## cond1c

  under_epi <- which(hz$hzdepb > diag$epipedon$bottom_depth) # epipedon could
extend to bottom of pedon?
  if(length(under_epi)!=0) {
    under_epi <- min(under_epi)
  }

  sl_1c <- prof_slice(minsurf+50,minsurf,hz)
```

```
  if(any(hz$redoxconcen[under_epi],sl_1c$redoxconcen)) {

    cond1c1 <- any(hz$colorchroma_d <= 2 | sl_1c$colorchroma_d <= 2)

  } else {

    cond1c1 <- FALSE

  }

  cond1c2 <- any(hz$colorchroma_d <= 1,na.rm=TRUE) | any(sl_1c$colorchroma_d
<= 1, na.rm=TRUE)


  if(any(cond1c1,cond1c2)) {

    cond1c <- TRUE

  } else {

    cond1c <- FALSE

  }


  ## cond1d

  cond1d <- any(sl_1c$aadipyridyl)


  ## cond2; groundwater not checked

  cond2 <- all(all((sl_1c$esp >= 15 | sl_1c$sar >= 13),na.rm=TRUE) &
           ( (hz$esp < sl_1c$esp[nrow(sl_1c)] & hz$hzdept >= minsurf+50) |
             (hz$sar < sl_1c$sar[nrow(sl_1c)] & hz$hzdept >= minsurf+50) ))


  tax[["aquepts"]]["cond1"][tax[["aquepts"]]["pedon"]==id,] <- cond1
  tax[["aquepts"]]["cond1a"][tax[["aquepts"]]["pedon"]==id,] <- cond1a
  tax[["aquepts"]]["cond1b"][tax[["aquepts"]]["pedon"]==id,] <- cond1b
  tax[["aquepts"]]["cond1c"][tax[["aquepts"]]["pedon"]==id,] <- cond1c
  tax[["aquepts"]]["cond1c1"][tax[["aquepts"]]["pedon"]==id,] <- cond1c1
  tax[["aquepts"]]["cond1c2"][tax[["aquepts"]]["pedon"]==id,] <- cond1c2
  tax[["aquepts"]]["cond1d"][tax[["aquepts"]]["pedon"]==id,] <- cond1d
  tax[["aquepts"]]["cond2"][tax[["aquepts"]]["pedon"]==id,] <- cond2


  if(cond1 & any(cond1a,cond1b,cond1c,cond1d) | cond2) {

    tax[["aquepts"]]["tax"][tax[["aquepts"]]["pedon"]==id,] <- TRUE

  } else {
```

```
    tax[["aquepts"]]["tax"][tax[["aquepts"]]["pedon"]==id,] <- FALSE

  }


  return(tax)


}

inceptisols.suborders.gelepts <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  cond <- diag$str$str == "gelic"

  if(cond) {

    tax[["gelepts"]]["tax"][tax[["gelepts"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["gelepts"]]["tax"][tax[["gelepts"]]["pedon"]==id,] <- FALSE

  }


  return(tax)

}

inceptisols.suborders.cryepts <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  cond <- diag$str$str == "cryic"

  if(cond) {

    tax[["cryepts"]]["tax"][tax[["cryepts"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["cryepts"]]["tax"][tax[["cryepts"]]["pedon"]==id,] <- FALSE

  }


  return(tax)


}
```

```
inceptisols.suborders.ustepts <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  cond <- diag$smr$smr == "ustic"

  if(cond) {

    tax[["ustepts"]]["tax"][tax[["ustepts"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["ustepts"]]["tax"][tax[["ustepts"]]["pedon"]==id,] <- FALSE

  }


  return(tax)


}

inceptisols.suborders.xerepts <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  cond <- diag$smr$smr == "xeric"

  if(cond) {

    tax[["xerepts"]]["tax"][tax[["xerepts"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["xerepts"]]["tax"][tax[["xerepts"]]["pedon"]==id,] <- FALSE

  }


  return(tax)


}

inceptisols.suborders.udepts <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  suborders <- c("aquepts","gelepts","cryepts","ustepts","xerepts")
```

```
  if(any(as.logical(lapply(tax[suborders],function(x){x[["tax"]]
[which(x[["pedon"]]==id)]}))) {

    tax[["udepts"]]["tax"][tax[["udepts"]]["pedon"]==id,] <- FALSE

  } else {

    tax[["udepts"]]["tax"][tax[["udepts"]]["pedon"]==id,] <- TRUE

  }

  return(tax)

}




##############################################################################
# TODO:
# - SKIPPED
# - how to check water potential?
entisols.suborders.wassents <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  return(tax)

}

# TODO:
# - simplified to aquic conditions at certain depth
# - need to include more complex conditions
entisols.suborders.aquents <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  minsurf <- diag$orgmin$min_top

  sl <- prof_slice(minsurf+50,minsurf,hz)

  cond <- any(sl$aquic)

  if(cond) {

    tax[["aquents"]]["tax"][tax[["aquents"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["aquents"]]["tax"][tax[["aquents"]]["pedon"]==id,] <- FALSE
```

```
  }

  return(tax)

}

entisols.suborders.psamments <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  sl <-
prof_slice(diag$partctrl$pc_bottom_depth,diag$partctrl$pc_top_depth,hz)

  cond <- all(sl$fragtotvol < 35) &
          all(sl$texture %in% c("loamy fine sand","loamy sand","loamy coarse
sand",
                                "very fine sand","find sand","sand","coarse
sand"))

  if(all(cond)) {

    tax[["psamments"]]["tax"][tax[["psamments"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["psamments"]]["tax"][tax[["psamments"]]["pedon"]==id,] <- FALSE

  }


  return(tax)


}

# TODO:
# - cond1 and cond2 not checked; slopes not checked
entisols.suborders.fluvents <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  minsurf <- diag$orgmin$min_top

  cond0_1 <- diag_top_depth(c("densic","lithic","paralithic"),diag,id)

  if(!is.na(cond0_1)) {

    cond3b_contact <- min(cond0_1)
    cond0_1 <- min(cond0_1) > 25

  } else {
```

```
    cond3b_contact <- minsurf+125
    cond0_1 <- TRUE

  }

  cond0_2 <- sum(hz$hzthk[hz$humanTpAlt],na.rm=TRUE) < 50

  if(cond0_1 & cond0_2) {
    cond0 <- TRUE
  } else{
     cond0 <- FALSE
   }


  ## cond1 skipped

  ## cond2 skipped

  ## cond3

  if(max(hz$hzdepb) < 125+minsurf) {

    cond3a <- FALSE

  } else {

    cond3a <- hz$oc[which(hz$hzdepb >= 125+minsurf & hz$hzdept < 125+minsurf)]
>= 0.2

  }


  sl <- prof_slice(cond3b_contact,25,hz)

  if(nrow(sl)>1) {
    cond3b <- sl$oc[1] > sl$oc[nrow(sl)]
  } else {
    cond3b <- FALSE
  }


  if(cond3a | cond3b) {
    cond3 <- TRUE
  } else {
    cond3 <- FALSE
  }


  ## cond 4

  cond4a <- !diag$str$str %in% c("cryic","gelic")

  if(!cond4a) {
```

```
    cond4b1 <- all(!hz$gelic_mat)
    cond4b2 <- all(hz$vglass < 15) # slope not checked

  } else {

    cond4b1 <- FALSE
    cond4b2 <- FALSE

  }

  if(cond4b1 & cond4b2 ) {
    cond4b <- TRUE
  } else {
    cond4b <- FALSE
  }


  if(cond4a | cond4b) {
    cond4 <- TRUE
  } else {
    cond4 <- FALSE
  }


  tax[["fluvents"]]["cond3"][tax[["fluvents"]]["pedon"]==id,] <- cond3
  tax[["fluvents"]]["cond3a"][tax[["fluvents"]]["pedon"]==id,] <- cond3a
  tax[["fluvents"]]["cond3b"][tax[["fluvents"]]["pedon"]==id,] <- cond3b
  tax[["fluvents"]]["cond4"][tax[["fluvents"]]["pedon"]==id,] <- cond4
  tax[["fluvents"]]["cond4a"][tax[["fluvents"]]["pedon"]==id,] <- cond4a
  tax[["fluvents"]]["cond4b"][tax[["fluvents"]]["pedon"]==id,] <- cond4b
  tax[["fluvents"]]["cond4b1"][tax[["fluvents"]]["pedon"]==id,] <- cond4b1
  tax[["fluvents"]]["cond4b2"][tax[["fluvents"]]["pedon"]==id,] <- cond4b2



  if(all(cond0,cond3,cond4)) {

    tax[["fluvents"]]["tax"][tax[["fluvents"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["fluvents"]]["tax"][tax[["fluvents"]]["pedon"]==id,] <- FALSE

  }



  return(tax)

}

entisols.suborders.orthents <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
```

```
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  suborders <- c("wassents","aquents","psamments","fluvents")

  if(any(as.logical(lapply(tax[suborders],function(x){x[["tax"]]
[which(x[["pedon"]]==id)]})))) {

    tax[["orthents"]]["tax"][tax[["orthents"]]["pedon"]==id,] <- FALSE

  } else {

    tax[["orthents"]]["tax"][tax[["orthents"]]["pedon"]==id,] <- TRUE

  }

  return(tax)

}
```

Appendix K:

gelisols.greatgroups.R

```
# This script contains the functions for classifying great groups for
gelisols.

# TODO: OVERALL
# - check interpretation for cumulative saturation; any hz?
# - add glacic layer to diag functions

# - for taxonomy functions, diag needs to be diags and filtered like hzs



### Histels ###
#############################################################################
gelisols.greatgroups.folistels <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  cond <- all(hz$cumusat < 30)

  if(cond) {

    tax[["folistels"]]["tax"][tax[["folistels"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["folistels"]]["tax"][tax[["folistels"]]["pedon"]==id,] <- FALSE

  }


  return(tax)


}

gelisols.greatgroups.glacistels <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})


  cond0 <- any(hz$cumusat > 30)

  cond1 <- diag$glacic$hz & diag$glacic$top_depth <= 100

  ## cond2

  contacts <- min(diag_top_depth(c("densic","lithic","paralithic"),diag,id))

  if(is.na(contacts)) {

    shallower <- 50
```

```
  } else if(contacts < 50) {

    shallower <- contacts

  } else {

    shallower <- 50

  }

  sl <- prof_slice(shallower,0,hz)
  cond2 <- all(sl$sphgm_vol < 75)


  tax[["glacistels"]]["cond0"][tax[["glacistels"]]["pedon"]==id,] <- cond0
  tax[["glacistels"]]["cond1"][tax[["glacistels"]]["pedon"]==id,] <- cond1
  tax[["glacistels"]]["cond2"][tax[["glacistels"]]["pedon"]==id,] <- cond2


  if(all(cond0,cond1,cond2)) {

    tax[["glacistels"]]["tax"][tax[["glacistels"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["glacistels"]]["tax"][tax[["glacistels"]]["pedon"]==id,] <- FALSE

  }


  return(tax)


}

gelisols.greatgroups.fibristels <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})


  contacts <- min(diag_top_depth(c("densic","lithic","paralithic"),diag,id))

  if(is.na(contacts)) {

    shallower <- 50

  } else if(contacts < 50) {

    shallower <- contacts

  } else {
```

```
    shallower <- 50

  }

  sl <- prof_slice(shallower,0,hz)
  fib_thk <- sum(sl$hzthk[sl$hzname=="Oi"])
  hem_thk <- sum(sl$hzthk[sl$hzname=="Oe"])
  sap_thk <- sum(sl$hzthk[sl$hzname=="Oa"])

  cond <- (fib_thk > hem_thk & fib_thk > sap_thk)

  if(cond) {

    tax[["fibristels"]]["tax"][tax[["fibristels"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["fibristels"]]["tax"][tax[["fibristels"]]["pedon"]==id,] <- TRUE

  }


  return(tax)


}

gelisols.greatgroups.hemistels <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})


  contacts <- min(diag_top_depth(c("densic","lithic","paralithic"),diag,id))

  if(is.na(contacts)) {

    shallower <- 50

  } else if(contacts < 50) {

    shallower <- contacts

  } else {

    shallower <- 50

  }

  sl <- prof_slice(shallower,0,hz)
  fib_thk <- sum(sl$hzthk[sl$hzname=="Oi"])
  hem_thk <- sum(sl$hzthk[sl$hzname=="Oe"])
  sap_thk <- sum(sl$hzthk[sl$hzname=="Oa"])
```

```
  cond <- (hem_thk > fib_thk & hem_thk > sap_thk)

  if(cond) {

    tax[["fibristels"]]["tax"][tax[["fibristels"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["fibristels"]]["tax"][tax[["fibristels"]]["pedon"]==id,] <- TRUE

  }


  return(tax)

}

gelisols.greatgroups.sapristels <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  greatgroups <-
c("folistels","glacistels","fibristels","hemistels","sapristels")

  if(any(as.logical(lapply(tax[greatgroups],function(x){x[["tax"]]
[which(x[["pedon"]]==id)]})))) {

    tax[["sapristels"]]["tax"][tax[["sapristels"]]["pedon"]==id,] <- FALSE

  } else {

    tax[["sapristels"]]["tax"][tax[["sapristels"]]["pedon"]==id,] <- TRUE

  }

  return(tax)

}









### Turbels ###
##############################################################################
gelisols.greatgroups.histoturbels <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})
```

```
  sl <- prof_slice(50,0,hz)

  chk_30 <- sum(sl$hzthk[sl$desgnmaster == "O"]) / sum(hz$hzthk) >= 0.3
  chk_40 <- sum(sl$hzthk[sl$desgnmaster == "O" & sl$cumusat >= 30]) /
sum(sl$hzthk) >= 0.4

  if(chk_30 & chk_40) {

    tax[["histotrubels"]]["tax"][tax[["histotrubels"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["histoturbels"]]["tax"][tax[["histoturbels"]]["pedon"]==id,] <- FALSE

  }


  return(tax)

}

# TODO:
# - doesn't specify wet or dry chroma; assumed wet given aquic conditions
gelisols.greatgroups.aquiturbels <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  minsurf <- diag$orgmin$min_top

  sl <- prof_slice(minsurf+50,minsurf,hz)

  cond <- any(sl$aquic & sl$redoxdepl & sl$colorchroma_w <= 2)

  if(cond) {

    tax[["aquiturbels"]]["tax"][tax[["aquiturbels"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["aquiturbels"]]["tax"][tax[["aquiturbels"]]["pedon"]==id,] <- FALSE

  }


  return(tax)

}

gelisols.greatgroups.anhyturbels <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})
```

```
  cond <- diag$anhydrous_cond$prop

  if(cond) {

    tax[["anhyturbels"]]["tax"][tax[["anhyturbels"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["anhyturbels"]]["tax"][tax[["anhyturbels"]]["pedon"]==id,] <- FALSE

  }


  return(tax)

}

gelisols.greatgroups.molliturbels <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  cond <- diag$mollic$hz

  if(cond) {

    tax[["molliturbels"]]["tax"][tax[["molliturbels"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["molliturbels"]]["tax"][tax[["molliturbels"]]["pedon"]==id,] <- FALSE

  }


  return(tax)

}

gelisols.greatgroups.umbriturbels <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  cond <- diag$umbric$hz

  if(cond) {

    tax[["umbriturbels"]]["tax"][tax[["umbriturbels"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["umbriturbels"]]["tax"][tax[["umbriturbels"]]["pedon"]==id,] <- TRUE
```

```
  }


  return(tax)

}

gelisols.greatgroups.psammoturbels <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  sl <-
prof_slice(diag$partctrl$pc_bottom_depth,diag$partctrl$pc_top_depth,hz)

  cond <- sl$fragtotvol < 35 &
          any(sl$texture %in% c("loamy fine sand","loamy sand","loamy coarse
sand",
                                "very fine sand","fine sand","sand","coarse
sand"))

  if(cond) {

    tax[["psammoturbels"]]["tax"][tax[["psammoturbels"]]["pedon"]==id,] <-
TRUE

  } else {

    tax[["psammoturbels"]]["tax"][tax[["psammoturbels"]]["pedon"]==id,] <-
FALSE

  }


  return(tax)

}

gelisols.greatgroups.haploturbels <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  greatgroups <- c("histoturbels","aquiturbels","anhyturbels","molliturbels",
                   "umbriturbels","psammoturbels","haploturbels")

  if(any(as.logical(lapply(tax[greatgroups],function(x){x[["tax"]]
[which(x[["pedon"]]==id)]})))) {

    tax[["haploturbels"]]["tax"][tax[["haploturbels"]]["pedon"]==id,] <- FALSE

  } else {
```

```
    tax[["haploturbels"]]["tax"][tax[["haploturbels"]]["pedon"]==id,] <- TRUE

  }

  return(tax)

}









### Orthels ###
################################################################################
gelisols.greatgroups.historthels <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  cond <- diag$histic$hz

  if(cond) {

    tax[["historthels"]]["tax"][tax[["historthels"]]["pedon"]==id,] <- TRUE


  } else {

    tax[["historthels"]]["tax"][tax[["historthels"]]["pedon"]==id,] <- FALSE

  }


  return(tax)

}

gelisols.greatgroups.aquorthels <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  minsurf <- diag$orgmin$min_top

  sl <- prof_slice(minsurf+50,minsurf,hz)

  cond <- any(sl$aquic & sl$redoxdepl & sl$colorchroma_w <= 2)

  if(cond) {
```

```
    tax[["aquorthels"]]["tax"][tax[["aquorthels"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["aquorthels"]]["tax"][tax[["aquorthels"]]["pedon"]==id,] <- FALSE

  }


  return(tax)

}

gelisols.greatgroups.anhyorthels <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  cond <- diag$anhydrous_cond$prop

  if(cond) {

    tax[["anhyorthels"]]["tax"][tax[["anhyorthels"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["anhyorthels"]]["tax"][tax[["anhyorthels"]]["pedon"]==id,] <- TRUE

  }


  return(tax)

}

gelisols.greatgroups.mollorthels <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  cond <- diag$mollic$hz

  if(cond) {

    tax[["mollorthels"]]["tax"][tax[["mollorthels"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["mollorthels"]]["tax"][tax[["mollorthels"]]["pedon"]==id,] <- FALSE

  }
```

```
  return(tax)

}

gelisols.greatgroups.umbrorthels <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  cond <- diag$umbric$hz

  if(cond) {

    tax[["umbrorthels"]]["tax"][tax[["umbrorthels"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["umbrorthels"]]["tax"][tax[["umbrorthels"]]["pedon"]==id,] <- FALSE

  }


  return(tax)

}

gelisols.greatgroups.argiorthels <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  minsurf <- diag$orgmin$min_top

  if(diag$argillic$hz) {

    cond <- diag$argillic$top_depth <= minsurf + 100

  } else {

    cond <- FALSE

  }


  if(cond) {

    tax[["argiorthels"]]["tax"][tax[["argiorthels"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["argiorthels"]]["tax"][tax[["argiorthels"]]["pedon"]==id,] <- FALSE

  }
```

```
  return(tax)

}

gelisols.greatgroups.psammorthels <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  if(any(hz$hzname == "Ap")) {

    ap_depth <- max(hz$hzdepb[hz$name == "Ap"])

    if(ap_depth > 25) {

      deeper <- ap_depth

    } else {

      deeper <- 25

    }

  }

  sl1 <- prof_slice(max(hz$hzdepb),deeper,hz)
  sl2 <-
prof_slice(diag$partctrl$pc_bottom_depth,diag$partctrl$pc_top_depth,hz)

  cond <- all(sl1$fragtotvol < 35) &
          any(sl2$texture %in%
                c("loamy fine sand","loamy sand","loamy coarse sand",
                  "very fine sand","fine sand","sand","coarse sand"))


  if(cond) {

    tax[["psammorthels"]]["tax"][tax[["psammorthels"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["psammorthels"]]["tax"][tax[["psammorthels"]]["pedon"]==id,] <- FALSE

  }


  return(tax)


}

gelisols.greatgroups.haplorthels <- function(hzs, diags, tax, id) {
```

```
  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  greatgroups <- c("historthels","aquorthels","anhyorthels","mollorthels",
                   "umbrorthels","argiorthels","psammorthels","haplorthels")

  if(any(as.logical(lapply(tax[greatgroups],function(x){x[["tax"]]
[which(x[["pedon"]]==id)]})))) {

    tax[["haplorthels"]]["tax"][tax[["haplorthels"]]["pedon"]==id,] <- FALSE

  } else {

    tax[["haplorthels"]]["tax"][tax[["haplorthels"]]["pedon"]==id,] <- TRUE

  }

  return(tax)

}
```

Appendix L:

histosols.greatgroups.R

```
# This script contains the functions for classifying great groups for
histosols.

# TODO: OVERALL
# - many check for sulfidic materials, need to add this to diag or hz structs

# - for taxonomy functions, diag needs to be diags and filtered like hzs



### Fibrists ###
################################################################################
histosols.greatgroups.cryofibrists <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  cond <- diag$str$str == "cryic"

  if(cond) {

    tax[["cryofibrists"]]["tax"][tax[["cryofibrists"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["cryofibrists"]]["tax"][tax[["cryofibrists"]]["pedon"]==id,] <- FALSE

  }


  return(tax)

}

# TODO:
# - normal check the contact; not needed here; check for mineral
# - deeper than 25cm since that is the allowed Ap depth for Histic epi
histosols.greatgroups.sphagnofibrists <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  if(any(hz$desgnmaster != "O" & hz$hzdepb > 25 & hz$hzdept < 90)) {

    shallower <- min(hz$hzdept[hz$desgnmaster != "O" & hz$hzdepb > 25 &
hz$hzdept < 90])

  } else {

    shallower <- 90

  }

  sl <- prof_slice(shallower,0,hz)
```

```
  cond <- sum((sl$sphgm_vol/100)*sl$hzthk) >= 0.75*sum(sl$hzthk)

  if(cond) {

    tax[["sphagnofibrists"]]["tax"][tax[["sphagnofibrists"]]["pedon"]==id,] <-
TRUE

  } else {

    tax[["sphagnofibrists"]]["tax"][tax[["sphagnofibrists"]]["pedon"]==id,] <-
FALSE

  }


  return(tax)

}

histosols.greatgroups.haplofibrists <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  greatgroups <- c("cryofibrists","sphagnofibrists")

  if(any(as.logical(lapply(tax[greatgroups],function(x){x[["tax"]]
[which(x[["pedon"]]==id)]})))) {

    tax[["haplofibrists"]]["tax"][tax[["haplofibrists"]]["pedon"]==id,] <-
FALSE

  } else {

    tax[["haplofibrists"]]["tax"][tax[["haplofibrists"]]["pedon"]==id,] <-
TRUE

  }

  return(tax)

}
```

```
### Folists ###
##############################################################################
```

```
histosols.greatgroups.cryofolists <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  cond <- diag$str$str == "cryic"

  if(cond) {

    tax[["cryofolists"]]["tax"][tax[["cryofolists"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["cryofolists"]]["tax"][tax[["cryofolists"]]["pedon"]==id,] <- FALSE

  }


  return(tax)

}

histosols.greatgroups.torrifolists <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  cond <- any(diag$str$str %in% c("aridic","torric"))

  if(cond) {

    tax[["torrifolists"]]["tax"][tax[["torrifolists"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["torrifolists"]]["tax"][tax[["torrifolists"]]["pedon"]==id,] <- FALSE

  }


  return(tax)

}

histosols.greatgroups.ustifolists <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  cond <- any(diag$smr$smr %in% c("ustic","xeric"))

  if(cond) {

    tax[["ustifolists"]]["tax"][tax[["ustifolists"]]["pedon"]==id,] <- TRUE
```

```
  } else {

    tax[["ustifolists"]]["tax"][tax[["ustifolists"]]["pedon"]==id,] <- FALSE

  }


  return(tax)

}

histosols.greatgroups.udifolists <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  greatgroups <- c("cryofolists","torrifolists","ustifolists")

  if(any(as.logical(lapply(tax[greatgroups],function(x){x[["tax"]]
[which(x[["pedon"]]==id)]})))) {

    tax[["udifolists"]]["tax"][tax[["udifolists"]]["pedon"]==id,] <- FALSE

  } else {

    tax[["udifolists"]]["tax"][tax[["udifolists"]]["pedon"]==id,] <- TRUE

  }

  return(tax)

}









### Hemists ###
##############################################################################
histosols.greatgroups.sulfohemists <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  cond <- diag$sulfuric$top_depth <= 50

  if(cond) {
```

```
    tax[["sulfohemists"]]["tax"][tax[["sulfohemists"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["sulfohemists"]]["tax"][tax[["sulfohemists"]]["pedon"]==id,] <- FALSE

  }


  return(tax)

}

# TODO:
# - SKIPPED
# - need to add sulfidic material to diag or hz
histosols.greatgroups.sulfihemists <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  return(tax)

}

# TODO:
# - SKIPPED
# - need to add humiluvic material to diag or hz
histosols.greatgroups.luvihemists <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  return(tax)

}

histosols.greatgroups.cryohemists <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  cond <- diag$str$str == "cryic"

  if(cond) {

    tax[["cryohemists"]]["tax"][tax[["cryohemists"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["cryohemists"]]["tax"][tax[["cryohemists"]]["pedon"]==id,] <- FALSE

  }
```

```
  return(tax)

}

histosols.greatgroups.haplohemists <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  greatgroups <- c("sulfohemists","sulfihemists","luvihemists","cryohemists")

  if(any(as.logical(lapply(tax[greatgroups],function(x){x[["tax"]]
[which(x[["pedon"]]==id)]})))) {

    tax[["haplohemists"]]["tax"][tax[["haplohemists"]]["pedon"]==id,] <- FALSE

  } else {

    tax[["haplohemists"]]["tax"][tax[["haplohemists"]]["pedon"]==id,] <- TRUE

  }

  return(tax)

}









### Saprists ###
################################################################################
histosols.greatgroups.sulfosaprists <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})


  if(diag$sulfuric$hz) {

    cond <- diag$sulfuric$top_depth <= 50

  } else {

    cond <- FALSE

  }
```

```
  if(cond) {

    tax[["sulfosaprists"]]["tax"][tax[["sulfosaprists"]]["pedon"]==id,] <-
TRUE

  } else {

    tax[["sulfosaprists"]]["tax"][tax[["sulfosaprists"]]["pedon"]==id,] <-
FALSE

  }


  return(tax)

}

# TODO:
# - SKIPPED
# - need to add sulfidic material to diag or hz
histosols.greatgroups.sulfisaprists <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  return(tax)

}

histosols.greatgroups.cryosaprists <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  cond <- diag$str$str == "cryic"

  if(cond) {

    tax[["cryosaprists"]]["tax"][tax[["cryosaprists"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["cryosaprists"]]["tax"][tax[["cryosaprists"]]["pedon"]==id,] <- FALSE

  }


  return(tax)

}

histosols.greatgroups.haplosaprists <- function(hzs, diags, tax, id) {
```

```
  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  greatgroups <- c("sulfosaprists","sulfisaprists","cryosaprists")

  if(any(as.logical(lapply(tax[greatgroups],function(x){x[["tax"]]
[which(x[["pedon"]]==id)]})))) {

    tax[["haplosaprists"]]["tax"][tax[["haplosaprists"]]["pedon"]==id,] <-
FALSE

  } else {

    tax[["haplosaprists"]]["tax"][tax[["haplosaprists"]]["pedon"]==id,] <-
TRUE

  }

  return(tax)

}









### Wassists ###
################################################################################
# TODO:
# - very specfic EC requirement; units of dS/m
histosols.greatgroups.frasiwassists <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  cond <- all(hz$ec_15 <= 0.2 & hz$hzdept < 100)

  if(cond) {

    tax[["frasiwassists"]]["tax"][tax[["frasiwassists"]]["pedon"]==id,] <-
TRUE

  } else {

    tax[["frasiwassists"]]["tax"][tax[["frasiwassists"]]["pedon"]==id,] <-
FALSE

  }
```

```
  return(tax)

}

# TODO:
# - SKIPPED
# - need to add sulfidic materials to diag or hz
histosols.greatgroups.sulfiwassists <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  return(tax)

}

histosols.greatgroups.haplowassists <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  greatgroups <- c("frasiwassists","sulfiwassists")

  if(any(as.logical(lapply(tax[greatgroups],function(x){x[["tax"]]
[which(x[["pedon"]]==id)]})))) {

    tax[["haplowassists"]]["tax"][tax[["haplowassists"]]["pedon"]==id,] <-
FALSE

  } else {

    tax[["haplowassists"]]["tax"][tax[["haplowassists"]]["pedon"]==id,] <-
TRUE

  }

  return(tax)


}
```

Appendix M:

alfisols.greatgroups.R

```
# This script contains the functions for classifying great groups for
alfisols.

# TODO:
# - for taxonomy functions, diag needs to be diags and filtered like hzs



### Aqualfs ###
##############################################################################
alfisols.greatgroups.cryaqualfs <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  cond <- diag$str$str == "cryic"

  if(cond) {

    tax[["cryaqualfs"]]["tax"][tax[["cryaqualfs"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["cryaqualfs"]]["tax"][tax[["cryaqualfs"]]["pedon"]==id,] <- FALSE

  }


  return(tax)

}

alfisols.greatgroups.plinthaqualfs <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  minsurf <- diag$orgmin$min_top

  sl <- prof_slice(minsurf+150,minsurf+30,hz)

  cond <- any(sl$plinthite_vol >= 50)

  if(cond) {

    tax[["plinthaqualfs"]]["tax"][tax[["plinthaqualfs"]]["pedon"]==id,] <-
TRUE

  } else {

    tax[["plinthaqualfs"]]["tax"][tax[["plinthaqualfs"]]["pedon"]==id,] <-
FALSE

  }
```

```
  return(tax)

}

alfisols.greatgroups.duraqualfs <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  cond <- diag$duripan$hz

  if(cond) {

    tax[["duraqualfs"]]["tax"][tax[["duraqualfs"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["duraqualfs"]]["tax"][tax[["duraqualfs"]]["pedon"]==id,] <- FALSE

  }


  return(tax)

}

alfisols.greatgroups.natraqualfs <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  cond <- diag$natric$hz

  if(cond) {

    tax[["natraqualfs"]]["tax"][tax[["natraqualfs"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["natraqualfs"]]["tax"][tax[["natraqualfs"]]["pedon"]==id,] <- FALSE

  }


  return(tax)

}

alfisols.greatgroups.fragiaqualfs <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})
```

```
  minsurf <- diag$orgmin$min_top

  if(diag$fragipan$hz) {

    cond <- diag$fragipan$top_depth <= minsurf+100

  } else {

    cond <- FALSE

  }


  if(cond) {

    tax[["fragiaqualfs"]]["tax"][tax[["fragiaqualfs"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["fragiaqualfs"]]["tax"][tax[["fragiaqualfs"]]["pedon"]==id,] <- FALSE

  }


  return(tax)

}

alfisols.greatgroups.kandiaqualfs <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  cond <- diag$kandic$hz

  if(cond) {

    tax[["kandiaqualfs"]]["tax"][tax[["kandiaqualfs"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["kandiaqualfs"]]["tax"][tax[["kandiaqualfs"]]["pedon"]==id,] <- FALSE

  }


  return(tax)

}

alfisols.greatgroups.vermaqualfs <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
```

```
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  minsurf <- diag$orgmin$min_top

  cond <- any(hz$hzdepb <= minsurf+100 & hz$hzthk >= 25 & hz$bioturb_vol >=
50)

  if(cond) {

    tax[["vermaqualfs"]]["tax"][tax[["vermaqualfs"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["vermaqualfs"]]["tax"][tax[["vermaqualfs"]]["pedon"]==id,] <- FALSE

  }


  return(tax)

}

# TODO:
# - mentions only the argillic hz
# - must make sure ksat units um/sec
alfisols.greatgroups.albaqualfs <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})


  ## check for abrupt textural change
  if(!diag$abrupt_text$prop) {

    tax[["albaqualfs"]]["tax"][tax[["albaqualfs"]]["pedon"]==id,] <- FALSE

    return(tax)

  }

  abrupt_top <- diag$abrupt_text$top_depth


  if(diag$argillic$hz) {

    arg_top <- diag$argillic$top_depth
    arg_bot <- diag$argillic$bottom_depth

    arg_sl <- prof_slice(arg_bot,arg_top,hz)

    if(any(is.na(arg_sl$ksat))) {
      cond_ksat <- FALSE
    } else {
      cond_ksat <- all(arg_sl$ksat <= 1.0)
```

```
    }

  } else {

    tax[["albaqualfs"]]["tax"][tax[["albaqualfs"]]["pedon"]==id,] <- FALSE

    return(tax)

  }


  if(diag$ochric$hz) {

    ochric <- diag$ochric$bottom_depth
    cond_ochric <- abrupt_top >= ochric & abrupt_top <= arg_bot

  } else {

    cond_ochric <- FALSE

  }

  if(diag$albic$hz) {

    albic <- diag$albic$bottom_depth
    cond_albic <- abrupt_top >= albic & abrupt_top <= arg_bot

  } else {

    cond_albic <- FALSE

  }


  if(cond_ksat & any(cond_ochric,cond_albic)) {

    tax[["albaqualfs"]]["tax"][tax[["albaqualfs"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["albaqualfs"]]["tax"][tax[["albaqualfs"]]["pedon"]==id,] <- FALSE

  }


  return(tax)

}

alfisols.greatgroups.glossaqualfs <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
```

```
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  cond <- diag$glossic$hz

  if(cond) {

    tax[["glossaqualfs"]]["tax"][tax[["glossaqualfs"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["glossaqualfs"]]["tax"][tax[["glossaqualfs"]]["pedon"]==id,] <- FALSE

  }


  return(tax)

}

# TODO:
# - consider a different method to check for sat mechanism
alfisols.greatgroups.epiaqualfs <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  cond <- any(!is.na(hz$sat_mech) & hz$sat_mech == "EPISAT")

  if(cond) {

    tax[["epiaqualfs"]]["tax"][tax[["epiaqualfs"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["epiaqualfs"]]["tax"][tax[["epiaqualfs"]]["pedon"]==id,] <- FALSE

  }


  return(tax)

}

alfisols.greatgroups.endoaqualfs <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  greatgroups <- c("cryaqualfs","plinthaqualfs","duraqualfs","natraqualfs",
                   "fragiaqualfs","kandiaqualfs","vermaqualfs","albaqualfs",
                   "glossaqualfs","epiaqualfs")

  if(any(as.logical(lapply(tax[greatgroups],function(x){x[["tax"]]
[which(x[["pedon"]]==id)]})))) {
```

```
    tax[["endoaqualfs"]]["tax"][tax[["endoaqualfs"]]["pedon"]==id,] <- FALSE

  } else {

    tax[["endoaqualfs"]]["tax"][tax[["endoaqualfs"]]["pedon"]==id,] <- TRUE

  }

  return(tax)


}









### Cryalfs ###
##############################################################################
# TODO:
# - difficult to check for interfingering; checks for albic materials in last
hz of arg,nat,kan
# cond1b not checked unless hz column added
alfisols.greatgroups.palecryalfs <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  # this should always return a value for alfisols
  argkannat_top <- diag_top_depth(c("argillic","kandic","natric"),diag,id)
  argkannat_bot <- diag_top_depth(c("argillic","kandic","natric"),diag,id)

  # sanity check if checked without order
  if(is.na(argkannat_top)) {

    tax[["palecryalfs"]]["tax"][tax[["palecryalfs"]]["pedon"]==id,] <- FALSE

    return(tax)

  }

  minsurf <- diag$orgmin$min_top

  cond1a <- argkannat_top >= minsurf + 60

  ## cond1b not checked
```

```
  sl <- prof_slice(argkannat_top,0,hz)

  # checked if not coarser
  cond2 <- any(!sl$texture %in% c("loamy fine sand","very fine sand","fine
sand",
                                  "sand","coarse sand"))


  sl_diag <- prof_slice(argkannat_bot,argkannat_top,hz)

  cond3 <- diag$glossic$hz | sl_diag$albic_vol[nrow(sl_diag)] > 0

  if(all(cond1a)) {

    cond1 <- TRUE

  } else {

    cond1 <- FALSE

  }


  tax[["palecryalfs"]]["tax"][tax[["palecryalfs"]]["cond1a"]==id,] <- cond1a
  tax[["palecryalfs"]]["tax"][tax[["palecryalfs"]]["cond1"]==id,] <- cond1
  tax[["palecryalfs"]]["tax"][tax[["palecryalfs"]]["cond2"]==id,] <- cond2
  tax[["palecryalfs"]]["tax"][tax[["palecryalfs"]]["cond3"]==id,] <- cond3


  if(all(cond1,cond2,cond3)) {


    tax[["palecryalfs"]]["tax"][tax[["palecryalfs"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["palecryalfs"]]["tax"][tax[["palecryalfs"]]["pedon"]==id,] <- FALSE

  }


  return(tax)



}

alfisols.greatgroups.glossocryalfs <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  cond <- diag$glossic$hz
```

```
  if(cond) {

    tax[["glossocryalfs"]]["tax"][tax[["glossocryalfs"]]["pedon"]==id,] <-
TRUE

  } else {

    tax[["glossocryalfs"]]["tax"][tax[["glossocryalfs"]]["pedon"]==id,] <-
FALSE

  }


  return(tax)

}

alfisols.greatgroups.haplocryalfs <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  greatgroups <- c("palecryalfs","glossocryalfs","haplocryalfs")

  if(any(as.logical(lapply(tax[greatgroups],function(x){x[["tax"]]
[which(x[["pedon"]]==id)]})))) {

    tax[["haplocryalfs"]]["tax"][tax[["haplocryalfs"]]["pedon"]==id,] <- FALSE

  } else {

    tax[["haplocryalfs"]]["tax"][tax[["haplocryalfs"]]["pedon"]==id,] <- TRUE

  }

  return(tax)

}









### Udalfs ###
################################################################################
alfisols.greatgroups.natrudalfs <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})
```

```
  cond <- diag$natric$hz

  if(cond) {

    tax[["natrudalfs"]]["tax"][tax[["natrudalfs"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["natrudalfs"]]["tax"][tax[["natrudalfs"]]["pedon"]==id,] <- FALSE

  }


  return(tax)

}

# TODO:
# - SKIPPED
alfisols.greatgroups.ferrudalfs <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  return(tax)

}

alfisols.greatgroups.fraglossudalfs <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  minsurf <- diag$orgmin$min_top

  cond1 <- diag$glossic$hz


  if(diag$fragipan$hz) {

    cond2 <- diag$fragipan$top_depth <= minsurf + 100

  } else {

    cond2 <- FALSE

  }



  tax[["fraglossudalfs"]]["cond1"][tax[["fraglossudalfs"]]["pedon"]==id,] <-
cond1
```

```
  tax[["fraglossudalfs"]]["cond2"][tax[["fraglossudalfs"]]["pedon"]==id,] <-
cond2


  if(cond1 & cond2) {

    tax[["fraglossudalfs"]]["tax"][tax[["fraglossudalfs"]]["pedon"]==id,] <-
TRUE

  } else {

    tax[["fraglossudalfs"]]["tax"][tax[["fraglossudalfs"]]["pedon"]==id,] <-
FALSE

  }


  return(tax)

}

alfisols.greatgroups.fragiudalfs <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})


  minsurf <- diag$orgmin$min_top

  if(diag$fragipan$hz) {

    cond <- diag$fragipan$top_depth <= minsurf + 100

  } else {

    cond <- FALSE

  }


  if(cond) {

    tax[["fragiudalfs"]]["tax"][tax[["fragiudalfs"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["fragiudalfs"]]["tax"][tax[["fragiudalfs"]]["pedon"]==id,] <- FALSE

  }


  return(tax)

}
```

```
# TODO:
# - just checks claytotal
# - just hz just above and below
# - the clay part of cond3b checked
alfisols.greatgroups.kandiudalfs <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  minsurf <- diag$orgmin$min_top

  contacts <-
min(diag_top_depth(c("densic","lithic","paralithic","petroferric"),diag,id))

  cond1 <- is.na(contacts) | contacts > minsurf+150

  cond2 <- diag$kandic$hz

  clay20idx <- which(hz$claytotal <= max(hz$claytotal) * 0.8 & hz$hzdept <
minsurf+150) # idx

  if(any(clay20idx > which.max(hz$claytotal))) {

    cond3a <- FALSE
    hzidx <- clay20idx[which(clay20idx > which.max(hz$claytotal))]


    if(max(hzidx) == nrow(hz)) {

      cond3b <- FALSE

    } else {

      cond3b <- hz$claytotal[max(hzidx)+1] >= hz$claytotal[max(hzidx)]+3

    }


  } else {

    cond3a <- TRUE
    cond3b <- FALSE

  }




  if(any(cond3a, cond3b)) {

    cond3 <- TRUE

  } else {
```

```
    cond3 <- FALSE

  }

  tax[["kandiudalfs"]]["cond1"][tax[["kandiudalfs"]]["pedon"]==id,] <- cond1
  tax[["kandiudalfs"]]["cond2"][tax[["kandiudalfs"]]["pedon"]==id,] <- cond2
  tax[["kandiudalfs"]]["cond3"][tax[["kandiudalfs"]]["pedon"]==id,] <- cond3
  tax[["kandiudalfs"]]["cond3a"][tax[["kandiudalfs"]]["pedon"]==id,] <- cond3a
  tax[["kandiudalfs"]]["cond3b"][tax[["kandiudalfs"]]["pedon"]==id,] <- cond3b


  if(all(cond1,cond2,cond3)) {

    tax[["kandiudalfs"]]["tax"][tax[["kandiudalfs"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["kandiudalfs"]]["tax"][tax[["kandiudalfs"]]["pedon"]==id,] <- FALSE

  }


  return(tax)


}

alfisols.greatgroups.kanhapludalfs <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  cond <- diag$kandic$hz

  if(cond) {

    tax[["kanhapludalfs"]]["tax"][tax[["kanhapludalfs"]]["pedon"]==id,] <-
TRUE

  } else {

    tax[["kanhapludalfs"]]["tax"][tax[["kanhapludalfs"]]["pedon"]==id,] <-
FALSE

  }


  return(tax)

}

# TODO:
# - assumed dry colors
```

```
# - cond4a2 skipped; consider adding vitric col
alfisols.greatgroups.paleudalfs <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})


  ## cond1 AND cond2 (a OR b) AND (cond3 (a OR b OR c) OR cond4 (a (1 AND 2)
AND b AND c))


  minsurf <- diag$orgmin$min_top


  ## cond1
  contacts <- diag_top_depth(c("densic","lithic","paralithic"),diag,id)

  if(all(is.na(contacts))) {

    cond1 <- TRUE

  } else {

    cond1 <- !any(contacts <= minsurf+150)

  }


  ## cond2

  clay20idx <- which(hz$claytotal <= max(hz$claytotal) * 0.8 & hz$hzdept <
minsurf+150) # idx

  if(any(clay20idx > which.max(hz$claytotal))) {

    cond2a <- FALSE
    hzidx <- clay20idx[which(clay20idx > which.max(hz$claytotal))]

    if(max(hzidx) == nrow(hz)) {

      cond2b <- FALSE

    } else {

      cond2b <- hz$claytotal[max(hzidx)+1] >= hz$claytotal[max(hzidx)]+3

    }




  } else {
```

```
    cond2a <- TRUE
    cond2b <- FALSE

  }




  ## cond3

  cond3 <- diag$argillic$hz

  if(cond3) {

    ## cond3a: bottom half of arg, hue of 7.5YR or redder, chroma 5+
    arg_sl_bot <-
prof_slice(diag$argillic$bottom_depth,diag$argillic$top_depth+(0.5*diag$argillic$thick),hz
    arg_sl <-
prof_slice(diag$argillic$bottom_depth,diag$argillic$top_depth,hz)

    cond3a <- any(arg_sl_bot$colorhue_d %in%
c("7.5YR","5YR","2.5YR","10R","7.5R","5R","2.5R") &
                    arg_sl_bot$colorchroma_d >= 5)

    cond3b <- sum(arg_sl$hzthk[(arg_sl$colorhue_d %in%
c("2.5YR","10R","7.5R","5R","2.5R") &
                arg_sl$colorvalue_w <= 3 & arg_sl$colorvalue_d <= 4)]) >=
0.5*diag$argillic$thick


    if(any(arg_sl$redoxconcen)) {

      cond3c <- any(arg_sl$redoxcolorhue %in%
c("5YR","2.5YR","10R","7.5R","5R","2.5R") |
                    arg_sl$redoxcolorchroma >= 6,na.rm=TRUE)

    } else {

      cond3c <- FALSE

    }

  } else {

    cond3 <- FALSE
    cond3a <- FALSE
    cond3b <- FALSE
    cond3c <- FALSE

  }
```

```
  if(any(cond2a,cond2b)) {

    cond2 <- TRUE

  } else {

    cond2 <- FALSE

  }


  if(any(cond3a,cond3b,cond3c)) {

    cond3 <- TRUE

  } else {

    cond3 <- FALSE

  }




  ## cond4

  cond4 <- diag$str$str == "figid"

  if(cond3 & cond4) {

    cond4a1 <- diag$argillic$top_depth >= minsurf+60


    if(all(cond4a1)) {

      cond4 <- TRUE

    } else {

      cond4 <- FALSE

    }

    # checking if NOT coarser
    cond4b <- any(!hz$texture %in% c("loamy find sand","loamy sand","loamy
coarse sand",
                                     "very fine sand", "fine sand",
"sand","coarse sand"))

    cond4c <- diag$glossic$hz | arg_sl$albic_vol[nrow(arg_sl)] > 0
```

```
  } else {

    cond4a <- FALSE
    cond4a1 <- FALSE
    # cond4a2 <- FALSE
    cond4b <- FALSE
    cond4c <- FALSE

  }


  if(all(cond4a,cond4b,cond4c)) {

    cond4 <- TRUE

  } else {

    cond4 <- FALSE

  }


  tax[["paleudalfs"]]["cond1"][tax[["paleudalfs"]]["pedon"]==id,] <- cond1
  tax[["paleudalfs"]]["cond2"][tax[["paleudalfs"]]["pedon"]==id,] <- cond2
  tax[["paleudalfs"]]["cond2a"][tax[["paleudalfs"]]["pedon"]==id,] <- cond2a
  tax[["paleudalfs"]]["cond2b"][tax[["paleudalfs"]]["pedon"]==id,] <- cond2b
  tax[["paleudalfs"]]["cond3"][tax[["paleudalfs"]]["pedon"]==id,] <- cond3
  tax[["paleudalfs"]]["cond3a"][tax[["paleudalfs"]]["pedon"]==id,] <- cond3a
  tax[["paleudalfs"]]["cond3b"][tax[["paleudalfs"]]["pedon"]==id,] <- cond3b
  tax[["paleudalfs"]]["cond3c"][tax[["paleudalfs"]]["pedon"]==id,] <- cond3c
  tax[["paleudalfs"]]["cond4"][tax[["paleudalfs"]]["pedon"]==id,] <- cond4
  tax[["paleudalfs"]]["cond4a"][tax[["paleudalfs"]]["pedon"]==id,] <- cond4a
  tax[["paleudalfs"]]["cond4a1"][tax[["paleudalfs"]]["pedon"]==id,] <- cond4a1
  tax[["paleudalfs"]]["cond4b"][tax[["paleudalfs"]]["pedon"]==id,] <- cond4b
  tax[["paleudalfs"]]["cond4c"][tax[["paleudalfs"]]["pedon"]==id,] <- cond4c


  if(cond1 & cond2 & (cond3 | cond4)) {

    tax[["paleudalfs"]]["tax"][tax[["paleudalfs"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["paleudalfs"]]["tax"][tax[["paleudalfs"]]["pedon"]==id,] <- FALSE

  }


  return(tax)


}
```

```
# TODO:
# - assumed dry colors when not specified
alfisols.greatgroups.rhodudalfs <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})


  cond0 <- diag$argillic$hz

  if(cond0) {

    if(diag$argillic$thick >= 100) {

      bot <- diag$argillic$top_depth + 100

    } else {

      bot <- diag$argillic$bottom_depth

    }

    sl <- prof_slice(bot, diag$argillic$top_depth, hz)

  } else {

    tax[["rhodudalfs"]]["cond0"][tax[["rhodudalfs"]]["pedon"]==id,] <- cond0
    tax[["rhodudalfs"]]["tax"][tax[["rhodudalfs"]]["pedon"]==id,] <- FALSE

    return(tax)

  }


  cond1 <- all(sl$colorhue_d %in% c("2.5YR","10R","7.5R","5R","2.5R"))

  cond2 <- all(sl$colorvalue_w <= 3)

  cond3 <- all((sl$colorvalue_d - sl$colorvalue_w) <= 1)


  tax[["rhodudalfs"]]["cond0"][tax[["rhodudalfs"]]["pedon"]==id,] <- cond0
  tax[["rhodudalfs"]]["cond1"][tax[["rhodudalfs"]]["pedon"]==id,] <- cond1
  tax[["rhodudalfs"]]["cond2"][tax[["rhodudalfs"]]["pedon"]==id,] <- cond2
  tax[["rhodudalfs"]]["cond3"][tax[["rhodudalfs"]]["pedon"]==id,] <- cond3

  if(all(cond0,cond1,cond2,cond3)) {

    tax[["rhodudalfs"]]["tax"][tax[["rhodudalfs"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["rhodudalfs"]]["tax"][tax[["rhodudalfs"]]["pedon"]==id,] <- FALSE
```

```
  }


  return(tax)


}

alfisols.greatgroups.glossudalfs <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  cond <- diag$glossic$hz

  if(cond) {

    tax[["glossudalfs"]]["tax"][tax[["glossudalfs"]]["pedon"]==id,] <- TRUE

  } else{

    tax[["glossudalfs"]]["tax"][tax[["glossudalfs"]]["pedon"]==id,] <- FALSE

  }


  return(tax)

}

alfisols.greatgroups.hapludalfs <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  greatgroups <-
c("natrudalfs","ferrudalfs","fraglossudalfs","fragiudalfs","kandiudalfs",
                  "kanhapludalfs","paleudalfs","rhodudalfs","glossudalfs")

  if(any(as.logical(lapply(tax[greatgroups],function(x){x[["tax"]]
[which(x[["pedon"]]==id)]})))) {

    tax[["hapludalfs"]]["tax"][tax[["hapludalfs"]]["pedon"]==id,] <- FALSE

  } else {

    tax[["hapludalfs"]]["tax"][tax[["hapludalfs"]]["pedon"]==id,] <- TRUE

  }

  return(tax)

}
```

```
### Ustalfs ###
################################################################################
# TODO:
# - first function with simplified true/false setting
alfisols.greatgroups.durustalfs <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  minsurf <- diag$orgmin$min_top

  if(diag$duripan$hz) {

    cond <- diag$duripan$top_depth <= minsurf + 100

  } else {

    cond <- FALSE

  }


  tax[["durustalfs"]]["tax"][tax[["durustalfs"]]["pedon"]==id,] <- cond

  return(tax)

}

alfisols.greatgroups.plinthustalfs <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  minsurf <- diag$orgmin$min_top

  cond <- any(hz$plinthite_vol >= 50 & hz$hzdept < minsurf + 150)

  tax[["plinthustalfs"]]["tax"][tax[["plinthustalfs"]]["pedon"]==id,] <- cond

  return(tax)

}

alfisols.greatgroups.natrustalfs <- function(hzs, diags, tax, id) {
```

```
  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  cond <- diag$natric$hz

  tax[["natrustalfs"]]["tax"][tax[["natrustalfs"]]["pedon"]==id,] <- cond

  return(tax)

}

alfisols.greatgroups.kandiustalfs <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  minsurf <- diag$orgmin$min_top

  contacts <-
min(diag_top_depth(c("densic","lithic","paralithic","petroferric"),diag,id))

  cond2 <- is.na(contacts) | contacts > minsurf+150

  cond1 <- diag$kandic$hz

  clay20idx <- which(hz$claytotal <= max(hz$claytotal) * 0.8 & hz$hzdept <
minsurf+150) # idx

  if(any(clay20idx > which.max(hz$claytotal))) {

    cond3a <- FALSE
    hzidx <- clay20idx[which(clay20idx > which.max(hz$claytotal))]

    if(max(hzidx) == nrow(hz)) {

    } else {

      cond3b <- hz$claytotal[max(hzidx)+1] >= hz$claytotal[max(hzidx)]+3

    }


  } else {

    cond3a <- TRUE
    cond3b <- FALSE

  }



  if(any(cond3a, cond3b)) {
```

```
    cond3 <- TRUE

  } else {

    cond3 <- FALSE

  }

  tax[["kandiustalfs"]]["cond1"][tax[["kandiustalfs"]]["pedon"]==id,] <- cond1
  tax[["kandiustalfs"]]["cond2"][tax[["kandiustalfs"]]["pedon"]==id,] <- cond2
  tax[["kandiustalfs"]]["cond3"][tax[["kandiustalfs"]]["pedon"]==id,] <- cond3
  tax[["kandiustalfs"]]["cond3a"][tax[["kandiustalfs"]]["pedon"]==id,] <-
cond3a
  tax[["kandiustalfs"]]["cond3b"][tax[["kandiustalfs"]]["pedon"]==id,] <-
cond3b


  if(all(cond1,cond2,cond3)) {

    tax[["kandiustalfs"]]["tax"][tax[["kandiustalfs"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["kandiustalfs"]]["tax"][tax[["kandiustalfs"]]["pedon"]==id,] <- FALSE

  }


  return(tax)

}

alfisols.greatgroups.kanhaplustalfs <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  cond <- diag$kandic$hz

  tax[["kanhaplustalfs"]]["tax"][tax[["kanhaplustalfs"]]["pedon"]==id,] <-
cond

  return(tax)

}

# TODO:
# - colors assumed dry when not specified
# - interpreted cond3a as just the first hz of arg from above arg
alfisols.greatgroups.paleustalfs <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})
```

```
  minsurf <- diag$orgmin$min_top


  ## cond1 OR cond2 (a (1 OR 2) AND b (1 OR 2)) OR cond3 (a OR b)

  ## cond1
  if(diag$petrocalcic$hz) {

    cond1 <- diag$petrocalcic$top_depth <= minsurf + 150

  } else {

    cond1 <- FALSE

  }



  ## cond2
  contacts <- diag_top_depth(c("densic","lithic","paralithic"),diag,id)

  if(diag$argillic$hz) {


    arg_sl <-
prof_slice(diag$argillic$bottom_depth,diag$argillic$top_depth,hz)


    if(is.na(contacts)) {

      cond2 <- TRUE

    } else {

      cond2 <- !any(contacts <= minsurf+150)

    }


    if(cond2) {


      ## cond2a

      clay20idx <- which(arg_sl$claytotal <= max(arg_sl$claytotal) * 0.8 &
arg_sl$hzdept < minsurf+150) # idx

      if(any(clay20idx > which.max(arg_sl$claytotal))) {

        cond2a1 <- FALSE
        hzidx <- clay20idx[which(clay20idx > which.max(arg_sl$claytotal))]


        if(max(hzidx) == nrow(hz)) {
```

```
          cond2a2 <- FALSE

        } else {

          cond2a2 <- arg_sl$claytotal[max(hzidx)+1] >=
arg_sl$claytotal[max(hzidx)]+3

        }


      } else {

        cond2a1 <- TRUE
        cond2a2 <- FALSE

      }




      ## cond2b

      arg_sl_bot <-
prof_slice(diag$argillic$bottom_depth,diag$argillic$top_depth+(0.5*diag$argillic$thick),hz

      cond2b1 <- any(arg_sl_bot$colorhue_d %in%
c("7.5YR","5YR","2.5YR","10R","7.5R","5R","2.5R") &
                       arg_sl_bot$colorchroma_d >= 5)

      cond2b2 <- any(arg_sl_bot$redoxconcen &
                       (arg_sl_bot$colorhue_d %in%
c("7.5YR","5YR","2.5YR","10R","7.5R","5R","2.5R"))|
                       (arg_sl_bot$colorchroma_d >= 6))


    } else {

      cond2a <- FALSE
      cond2a1 <- FALSE
      cond2a2 <- FALSE
      cond2b <- FALSE
      cond2b1 <- FALSE
      cond2b2 <- FALSE

    }



    ## cond3

    # using the top half as the "upper part"
    arg_sl_top <-
prof_slice(diag$argillic$top_depth+(0.5*diag$argillic$thick),diag$argillic$top_depth,hz)
```

```
    # better way to do the contacts check

    if(is.na(contacts)) {

      cond3 <- any(arg_sl_top$claytotal >= 35)

    } else {

      cond3 <- !any(contacts <= minsurf+50) & any(arg_sl_top$claytotal >= 35)

    }

    cond3a_sl_75 <-
prof_slice(diag$argillic$top_depth,diag$argillic$top_depth-7.5,hz)
    cond3a_sl_25 <-
prof_slice(diag$argillic$top_depth,diag$argillic$top_depth-2.5,hz)
    cond3a <- any(arg_sl_top$claytotal[1] - cond3a_sl_75$claytotal >= 20) |
      any(arg_sl_top$claytotal[1] - cond3a_sl_25$claytotal >= 15)

    # assumes nearest without 't' is eluvial
    el_idx <- which.max(!grepl(as.character(hz$hzname), pattern = "[Tt]")
                        & hz$hzdept > diag$argillic$top_depth)
    cond3b <- diag$abrupt_text$prop &
              (diag$abrupt_text$top_depth > hz$hzdept[el_idx] &
                 diag$abrupt_text$top_depth <= diag$argillic$top_depth)



  } else {

    cond2 <- FALSE
    cond2a <- FALSE
    cond2a1 <- FALSE
    cond2a2 <- FALSE
    cond2b <- FALSE
    cond2b1 <- FALSE
    cond2b2 <- FALSE
    cond3 <- FALSE
    cond3a <- FALSE
    cond3b <- FALSE

  }



  if(all(cond2,cond2a1,cond2a2)) {

    cond2a <- TRUE

  } else {

    cond2a <- FALSE
```

```
  }

  if(all(cond2,cond2b1,cond2b2)) {

    cond2b <- TRUE

  } else {

    cond2b <- FALSE

  }


  if(cond2a & cond2b) {

    cond2 <- TRUE

  } else {

    cond2 <- FALSE

  }




  if(all(cond3,cond3a,cond3b)) {

    cond3 <- TRUE

  } else {

    cond3 <- FALSE

  }


  tax[["paleustalfs"]]["cond1"][tax[["paleustalfs"]]["pedon"]==id,] <- cond1
  tax[["paleustalfs"]]["cond2"][tax[["paleustalfs"]]["pedon"]==id,] <- cond2
  tax[["paleustalfs"]]["cond2a"][tax[["paleustalfs"]]["pedon"]==id,] <- cond2a
  tax[["paleustalfs"]]["cond2a1"][tax[["paleustalfs"]]["pedon"]==id,] <-
cond2a1
  tax[["paleustalfs"]]["cond2a2"][tax[["paleustalfs"]]["pedon"]==id,] <-
cond2a2
  tax[["paleustalfs"]]["cond2b"][tax[["paleustalfs"]]["pedon"]==id,] <- cond2b
  tax[["paleustalfs"]]["cond2b1"][tax[["paleustalfs"]]["pedon"]==id,] <-
cond2b1
  tax[["paleustalfs"]]["cond2b2"][tax[["paleustalfs"]]["pedon"]==id,] <-
cond2b2
  tax[["paleustalfs"]]["cond3"][tax[["paleustalfs"]]["pedon"]==id,] <- cond3
  tax[["paleustalfs"]]["cond3a"][tax[["paleustalfs"]]["pedon"]==id,] <- cond3a
  tax[["paleustalfs"]]["cond3b"][tax[["paleustalfs"]]["pedon"]==id,] <- cond3b
```

```
  tax[["paleustalfs"]]["tax"][tax[["paleustalfs"]]["pedon"]==id,] <-
any(cond1,cond2,cond3)


  return(tax)



}

alfisols.greatgroups.rhodustalfs <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  cond0 <- diag$argillic$hz

  if(cond0) {

    if(diag$argillic$thick >= 100) {

      bot <- diag$argillic$top_depth + 100

    } else {

      bot <- diag$argillic$bottom_depth

    }

    sl <- prof_slice(bot, diag$argillic$top_depth, hz)

  } else {

    tax[["rhodustalfs"]]["cond0"][tax[["rhodustalfs"]]["pedon"]==id,] <- cond0
    tax[["rhodustalfs"]]["tax"][tax[["rhodustalfs"]]["pedon"]==id,] <- FALSE

    return(tax)

  }


  cond1 <- all(sl$colorhue_d %in% c("2.5YR","10R","7.5R","5R","2.5R"))

  cond2 <- all(sl$colorvalue_w <= 3)

  cond3 <- all((sl$colorvalue_d - sl$colorvalue_w) <= 1)


  tax[["rhodustalfs"]]["cond0"][tax[["rhodustalfs"]]["pedon"]==id,] <- cond0
  tax[["rhodustalfs"]]["cond1"][tax[["rhodustalfs"]]["pedon"]==id,] <- cond1
  tax[["rhodustalfs"]]["cond2"][tax[["rhodustalfs"]]["pedon"]==id,] <- cond2
  tax[["rhodustalfs"]]["cond3"][tax[["rhodustalfs"]]["pedon"]==id,] <- cond3

  if(all(cond0,cond1,cond2,cond3)) {
```

```
    tax[["rhodustalfs"]]["tax"][tax[["rhodustalfs"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["rhodustalfs"]]["tax"][tax[["rhodustalfs"]]["pedon"]==id,] <- FALSE

  }


  return(tax)

}

alfisols.greatgroups.haplustalfs <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  greatgroups <-
c("durustalfs","plinthustalfs","natrustalfs","kandiustalfs","kanhaplustalfs")

  if(any(as.logical(lapply(tax[greatgroups],function(x){x[["tax"]]
[which(x[["pedon"]]==id)]})))) {

    tax[["haplustalfs"]]["tax"][tax[["haplustalfs"]]["pedon"]==id,] <- FALSE

  } else {

    tax[["haplustalfs"]]["tax"][tax[["haplustalfs"]]["pedon"]==id,] <- TRUE

  }

  return(tax)

}









### Xeralfs ###
##############################################################################
alfisols.greatgroups.durixeralfs <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  minsurf <- diag$orgmin$min_top
```

```
  if(diag$duripan$hz) {

    cond <- diag$duripan$top_depth <= minsurf + 100

  } else {

    cond <- FALSE

  }

  tax[["durixeralfs"]]["tax"][tax[["durixeralfs"]]["pedon"]==id,] <- cond

  return(tax)

}

alfisols.greatgroups.natrixeralfs <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  cond <- diag$natric$hz

  tax[["natrixeralfs"]]["tax"][tax[["natrixeralfs"]]["pedon"]==id,] <- cond

  return(tax)

}

alfisols.greatgroups.fragixeralfs <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  minsurf <- diag$orgmin$min_top

  if(diag$fragipan$hz) {

    cond <- diag$fragipan$top_depth <= minsurf + 100

  } else {

    cond <- FALSE

  }


  tax[["fragixeralfs"]]["tax"][tax[["fragixeralfs"]]["pedon"]==id,] <- cond

  return(tax)

}
```

```
alfisols.greatgroups.plinthoxeralfs <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  minsurf <- diag$orgmin$min_top

  cond <- any(hz$plinthite_vol >= 50 & hz$hzdept < minsurf + 150)

  tax[["plinthoxeralfs"]]["tax"][tax[["plinthoxeralfs"]]["pedon"]==id,] <-
cond

  return(tax)

}

alfisols.greatgroups.rhodoxeralfs <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})


  cond0 <- diag$argillic$hz | diag$kandic$hz

  if(cond0) {

    if(diag$argillic$hz) {

      top <- diag$argillic$top_depth


      if(diag$argillic$thick >= 100) {

        bot <- diag$argillic$top_depth + 100

      } else {

        bot <- diag$argillic$bottom_depth

      }

    } else if(diag$kandic$hz) {


    top <- diag$kandic$top_depth


    if(diag$kandic$thick >= 100) {

      bot <- diag$kandic$top_depth + 100

    } else {

      bot <- diag$kandic$bottom_depth
```

```
    }


    }


  } else {

    tax[["rhodoxeralfs"]]["cond0"][tax[["rhodoxeralfs"]]["pedon"]==id,] <-
cond0
    tax[["rhodoxeralfs"]]["tax"][tax[["rhodoxeralfs"]]["pedon"]==id,] <- FALSE

    return(tax)

  }


  sl <- prof_slice(bot, top, hz)


  cond1 <- all(sl$colorhue_d %in% c("2.5YR","10R","7.5R","5R","2.5R"))

  cond2 <- all(sl$colorvalue_w <= 3)

  cond3 <- all((sl$colorvalue_d - sl$colorvalue_w) <= 1)


  tax[["rhodoxeralfs"]]["cond0"][tax[["rhodoxeralfs"]]["pedon"]==id,] <- cond0
  tax[["rhodoxeralfs"]]["cond1"][tax[["rhodoxeralfs"]]["pedon"]==id,] <- cond1
  tax[["rhodoxeralfs"]]["cond2"][tax[["rhodoxeralfs"]]["pedon"]==id,] <- cond2
  tax[["rhodoxeralfs"]]["cond3"][tax[["rhodoxeralfs"]]["pedon"]==id,] <- cond3

  if(all(cond0,cond1,cond2,cond3)) {

    tax[["rhodoxeralfs"]]["tax"][tax[["rhodoxeralfs"]]["pedon"]==id,] <- TRUE

  } else {

    tax[["rhodoxeralfs"]]["tax"][tax[["rhodoxeralfs"]]["pedon"]==id,] <- FALSE

  }


  return(tax)

}

# TODO:
# - interpreted cond3a as just the first hz of arg/kand from above arg/kand
alfisols.greatgroups.palexeralfs <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})
```

```
  minsurf <- diag$orgmin$min_top


  ## cond1 OR cond2 (a (1 OR 2) AND b (1 OR 2)) OR cond3 (a OR b)

  ## cond1

  if(diag$petrocalcic$hz) {

    cond1 <- diag$petrocalcic$top_depth <= minsurf + 150

  } else {

    cond1 <- FALSE

  }



  ## cond2
  contacts <- diag_top_depth(c("densic","lithic","paralithic"),diag,id)

  if(diag$argillic$hz) {
    ## check for argillic


    arg_sl <-
prof_slice(diag$argillic$bottom_depth,diag$argillic$top_depth,hz)


    if(is.na(contacts)) {

      cond2 <- TRUE

    } else {

      cond2 <- any(contacts <= minsurf+150)

    }


    if(cond2) {


      ## cond2a

      clay20idx <- which(arg_sl$claytotal <= max(arg_sl$claytotal) * 0.8 &
arg_sl$hzdept < minsurf+150) # idx

      if(any(clay20idx > which.max(arg_sl$claytotal))) {

        cond2a1 <- FALSE
        hzidx <- clay20idx[which(clay20idx > which.max(arg_sl$claytotal))]
```

```
        if(max(hzidx) == nrow(hz)) {

          cond2a2 <- FALSE

        } else {

          cond2a2 <- arg_sl$claytotal[max(hzidx)+1] >=
arg_sl$claytotal[max(hzidx)]+3

        }


      } else {

        cond2a1 <- TRUE
        cond2a2 <- FALSE

      }




      ## cond2b

      cond2b <- diag$argillic$bottom_depth >= 150


    } else {

      cond2a <- FALSE
      cond2a1 <- FALSE
      cond2a2 <- FALSE
      cond2b <- FALSE

    }



    ## cond3

    # using the top half as the "upper part"
    arg_sl_top <-
prof_slice(diag$argillic$top_depth+(0.5*diag$argillic$thick),diag$argillic$top_depth,hz)

    # better way to do the contacts check
    if(is.na(contacts)) {

      cond3 <- any(arg_sl_top$claytotal >= 35)

    } else {

      cond3 <- !any(contacts <= minsurf+50) & any(arg_sl_top$claytotal >= 35)
```

```
    }


    cond3a_sl_75 <-
prof_slice(diag$argillic$top_depth,diag$argillic$top_depth-7.5,hz)
    cond3a_sl_25 <-
prof_slice(diag$argillic$top_depth,diag$argillic$top_depth-2.5,hz)
    cond3a <- any(arg_sl_top$claytotal[1] - cond3a_sl_75$claytotal >= 20) | #
checking clay increase at upper bound
      any(arg_sl_top$claytotal[1] - cond3a_sl_25$claytotal >= 15)


    # assumes nearest without 't' is eluvial
    el_idx <- max(which(!grepl(as.character(hz$hzname), pattern = "[Tt]") &
hz$hzdept < diag$argillic$top_depth))
    cond3b <- diag$abrupt_text$prop &
      (diag$abrupt_text$top_depth > hz$hzdept[el_idx] &
         diag$abrupt_text$top_depth <= diag$argillic$top_depth)



  } else if(diag$kandic$hz) {
    ## check for kandic

    kan_sl <- prof_slice(diag$kandic$bottom_depth,diag$kandic$top_depth,hz)


    if(is.na(contacts)) {

      cond2 <- TRUE

    } else {

      cond2 <- any(contacts <= minsurf+150)

    }


    if(cond2) {


      ## cond2a

      clay20idx <- which(kan_sl$claytotal <= max(kan_sl$claytotal) * 0.8 &
kan_sl$hzdept < minsurf+150) # idx

      if(any(clay20idx > which.max(arg_sl$claytotal))) {

        cond2a1 <- FALSE
        hzidx <- clay20idx[which(clay20idx > which.max(kan_sl$claytotal))]


        if(max(hzidx) == nrow(hz)) {
```

```
          cond2a2 <- FALSE

        } else {

          cond2a2 <- kan_sl$claytotal[max(hzidx)+1] >=
kan_sl$claytotal[max(hzidx)]+3

        }



      } else {

        cond2a1 <- TRUE
        cond2a2 <- FALSE

      }




      ## cond2b

      cond2b <- diag$kandic$bottom_depth >= 150


    } else {

      cond2a <- FALSE
      cond2a1 <- FALSE
      cond2a2 <- FALSE
      cond2b <- FALSE

    }



    ## cond3

    # using the top half as the "upper part"
    kan_sl_top <-
prof_slice(diag$kandic$top_depth+(0.5*diag$kandic$thick),diag$kandic$top_depth,hz)

    # better way to do the contacts check
    if(is.na(contacts)) {

      cond3 <- any(kan_sl_top$claytotal >= 35)

    } else {

      cond3 <- !any(contacts <= minsurf+50) & any(kan_sl_top$claytotal >= 35)

    }
```

```
    cond3a_sl_75 <-
prof_slice(diag$kandic$top_depth,diag$kandic$top_depth-7.5,hz)
    cond3a_sl_25 <-
prof_slice(diag$kandic$top_depth,diag$kandic$top_depth-2.5,hz)
    cond3a <- any(kan_sl_top$claytotal[1] - cond3a_sl_75$claytotal >= 20) |
      any(kan_sl_top$claytotal[1] - cond3a_sl_25$claytotal >= 15)

    # assumes nearest without 't' is eluvial
    el_idx <- which.max(!grepl(as.character(hz$hzname), pattern = "[Tt]")
                        & hz$hzdept > diag$kandic$top_depth)
    cond3b <- diag$abrupt_text$prop &
      (diag$abrupt_text$top_depth > hz$hzdept[el_idx] &
         diag$abrupt_text$top_depth <= diag$kandic$top_depth)



  } else {

    cond2 <- FALSE
    cond2a <- FALSE
    cond2a1 <- FALSE
    cond2a2 <- FALSE
    cond2b <- FALSE
    cond3 <- FALSE
    cond3a <- FALSE
    cond3b <- FALSE

  }







  if(cond2 & any(cond2a1,cond2a2,na.rm=TRUE)) {

    cond2a <- TRUE

  } else {

    cond2a <- FALSE

  }


  if(all(cond2,cond2a,cond2b)) {

    cond2 <- TRUE

  } else {

    cond2 <- FALSE
```

```
  }


  if(cond3 & (cond3a | cond3b)) {

    cond3 <- TRUE

  } else {

    cond3 <- FALSE

  }


  tax[["palexeralfs"]]["cond1"][tax[["palexeralfs"]]["pedon"]==id,] <- cond1
  tax[["palexeralfs"]]["cond2"][tax[["palexeralfs"]]["pedon"]==id,] <- cond2
  tax[["palexeralfs"]]["cond2a"][tax[["palexeralfs"]]["pedon"]==id,] <- cond2a
  tax[["palexeralfs"]]["cond2a1"][tax[["palexeralfs"]]["pedon"]==id,] <-
cond2a1
  tax[["palexeralfs"]]["cond2a2"][tax[["palexeralfs"]]["pedon"]==id,] <-
cond2a2
  tax[["palexeralfs"]]["cond2b"][tax[["palexeralfs"]]["pedon"]==id,] <- cond2b
  tax[["palexeralfs"]]["cond3"][tax[["palexeralfs"]]["pedon"]==id,] <- cond3
  tax[["palexeralfs"]]["cond3a"][tax[["palexeralfs"]]["pedon"]==id,] <- cond3a
  tax[["palexeralfs"]]["cond3b"][tax[["palexeralfs"]]["pedon"]==id,] <- cond3b


  tax[["palexeralfs"]]["tax"][tax[["palexeralfs"]]["pedon"]==id,] <-
any(cond1,cond2,cond3)


  return(tax)


}

alfisols.greatgroups.haploxeralfs <- function(hzs, diags, tax, id) {

  hz <- dplyr::filter(hzs, pedon_id==id)
  diag <- lapply(diags,function(x){dplyr::filter(x,pedon==id)})

  greatgroups <-
c("durixeralfs","natrixeralfs","fragixeralfs","plinthoxeralfs",
                  "rhodoxeralfs","palexeralfs")

  if(any(as.logical(lapply(tax[greatgroups],function(x){x[["tax"]]
[which(x[["pedon"]]==id)]})))) {

    tax[["haploxeralfs"]]["tax"][tax[["haploxeralfs"]]["pedon"]==id,] <- FALSE

  } else {

    tax[["haploxeralfs"]]["tax"][tax[["haploxeralfs"]]["pedon"]==id,] <- TRUE
```

```
  }

  return(tax)

}
```

Appendix N:

helpers.R

```
# This script contains all of the "helper" functions used by the other files.
#
#


diag_top_depth <- function(diag_hz,diag,pedon_id) {
  #' This function makes it easier to find the top depths of
  #' any given subsurface horizons.
  #'

  hz_true <- diag_hz[as.logical(lapply(diag[diag_hz],function(x){x[["hz"]]
[which(x[["pedon"]]==pedon_id)]}))]

  top_depths <- unlist(lapply(diag[hz_true],function(x){x[["top_depth"]]
[which(x[["pedon"]]==pedon_id)]}))
  top_depth <- top_depths[!is.na(top_depths)]


  if(length(top_depth)==0) {

    top_depth <- NA

  }

  return(top_depth)

}




diag_bot_depth <- function(diag_hz,diag,pedon_id) {
  #' This function makes it easier to find the top depths of
  #' any given subsurface horizons.
  #'

  hz_true <- diag_hz[as.logical(lapply(diag[diag_hz],function(x){x[["hz"]]
[which(x[["pedon"]]==pedon_id)]}))]

  bot_depths <- unlist(lapply(diag[hz_true],function(x){x[["bottom_depth"]]
[which(x[["pedon"]]==pedon_id)]}))
  bot_depth <- bot_depths[!is.na(bot_depths)]


  if(length(bot_depth)==0) {

    bot_depth <- NA

  }

  return(bot_depth)

}
```

```
prof_slice <- function(bottom,top,hz) {
  #' A base R function to slice horizon tables for a section of the profile
  #' @param hz horizon data table
  #' @param top top of the profile section
  #' @param bottom bottom of the profile section


  hz <- hz[order(hz$hzdept),]

  t <- mapply(
    FUN = seq,
    from = hz[["hzdept"]],
    to = hz[["hzdepb"]],
    by = 0.5,
    SIMPLIFY = FALSE
  )

  if(bottom > max(hz$hzdepb)) {
    bottom <- max(hz$hzdepb)
  }

  if(top > max(hz$hzdepb)) {

    stop('top of profile slice is below pedon max depth')

  }

  if(top < 0) {

    top <- 0

  }

  # hz index of top of slice
  sl1 <- which(sapply(t, function(e) is.element(top, e)))
  # hz index of bottom  of slice
  sl2 <- which(sapply(t, function(e) is.element(bottom, e)))

  if(length(sl1) > 1 & (top %in% hz[["hzdept"]])) {
    sl1 <- sl1[2]
  }

  if(length(sl2) > 1 & (bottom %in% hz[["hzdepb"]])) {
    sl2 <- sl2[2]
  }

  hz$hzdept[sl1] <- top
  hz$hzdepb[sl2] <- bottom
```

```
  slhz <- hz[c(sl1:sl2),]

  if(any(slhz$hzdept == slhz$hzdepb)) {
    slhz <- slhz[which(!slhz$hzdept == slhz$hzdepb),]
  }

  slhz$hzthk <- slhz$hzdepb - slhz$hzdept

  return(slhz)

}




roll_wa <- function(hz,s,prop) {
  #' Helper function that generates a vector of weighted means based on
  #' a range of thicknesses
  #' @param hz horizon data table
  #' @param s sequence of bottom depths
  #' @param prop property/column to apply weighted mean
  #' @return vector of weighted means same length as sequence

  pl <- lapply(s,prof_slice,top=1,hz=hz)

  v <- lapply(pl,function(y) {y[,prop]}) # values
  t <- lapply(pl,function(y) {y[,"hzdepb"] - y[,"hzdept"]}) # thicknesses
  m <- lapply(pl,function(y) {max(y[,"hzdepb"])})
  w <- Map('/',t,m) # weights
  wa <- mapply(weighted.mean,v,w) # weighted mean; vector

  return(wa)

}




is.sequential <- function(x){
  #' Helper function to check if indices are sequential
  #' @param x vector of integers
  #' @return boolean

  all(diff(x) == diff(x)[1])
}




get_seq <- function(vec){
  #' Returns the FIRST continuous sequence of integers
```

```
  if(length(vec)==1) {
    return(vec)
  }

  d1 <-  which(diff(vec) == 1)
  if(all(diff(d1) == 1)){
    ridx <- c(d1, d1[length(d1)]+1)
    return(vec[ridx])
  }else{
    d2 <- split(d1, cumsum(c(1, diff(d1) != 1)))[[1]]
    ridx <- c(d2, d2[length(d2)]+1)
    return(vec[ridx])
  }
}




crit.clay.argillic <- function(eluvial_clay_content) {
  # eluvial clay content is a numeric vector or matrix subsettable with
logical vectors based on clay (NA omitted)
  buf <- eluvial_clay_content
  idx.mask <- is.na(buf)
  buf[idx.mask] <- 0

  idx.lt15 <- buf < 15
  idx.lt15[idx.mask] <- FALSE

  idx.gt40 <- buf >= 40
  idx.gt40[idx.mask] <- FALSE

  idx.other <- (buf >= 15) & (buf < 40)
  idx.other[idx.mask] <- FALSE

  buf[idx.lt15] <- eluvial_clay_content[idx.lt15] + 3
  buf[idx.gt40] <- eluvial_clay_content[idx.gt40] + 8
  buf[idx.other] <- 1.2*eluvial_clay_content[idx.other]
  return(round(buf))
}

crit.clay.kandic <- function(eluvial_clay_content) {
  # eluvial clay content is a numeric vector or matrix subsettable with
logical vectors based on clay (NA omitted)
  buf <- eluvial_clay_content
  idx.mask <- is.na(buf)
  buf[idx.mask] <- 0

  idx.lt20 <- buf < 20
  idx.lt20[idx.mask] <- FALSE

  idx.gt40 <- buf >= 40
  idx.gt40[idx.mask] <- FALSE
```

```
  idx.other <- (buf >= 20) & (buf < 40)
  idx.other[idx.mask] <- FALSE

  buf[idx.lt20] <- eluvial_clay_content[idx.lt20] + 4
  buf[idx.gt40] <- eluvial_clay_content[idx.gt40] + 8
  buf[idx.other] <- 1.2*eluvial_clay_content[idx.other]
  return(round(buf))
}

crit.clay.oxic <- function(eluvial_clay_content) {
  # eluvial clay content is a numeric vector or matrix subsettable with
logical vectors based on clay (NA omitted)
  buf <- eluvial_clay_content
  idx.mask <- is.na(buf)
  buf[idx.mask] <- 0

  idx.lt20 <- buf < 20
  idx.lt20[idx.mask] <- FALSE

  idx.gt40 <- buf >= 40
  idx.gt40[idx.mask] <- FALSE

  idx.other <- (buf >= 20) & (buf < 40)
  idx.other[idx.mask] <- FALSE

  buf[idx.lt20] <- eluvial_clay_content[idx.lt20] + 4
  buf[idx.gt40] <- eluvial_clay_content[idx.gt40] + 8
  buf[idx.other] <- 1.2*eluvial_clay_content[idx.other]
  return(round(buf))
}



get.increase.matrix <- function(hz, attr, threshold.fun, vertical.distance) {
  # p - a SoilProfileCollection, containing a single profile
  # attr - attribute name to get the "increase" of
  # threshold.fun - a function that returns the threshold (as a function of
attr); may return a constant single value
  # vertical distance - the vertical distance (determined from difference SPC
top depth variable) within which increase must be met

  # NOTE: should topdepths or midpoints be used? typically the same horizon
index is chosen
  #
  #       with thick horizons/diffuse transitions -- the midpoint is probably
more appropriate.
  #
  #       I originally chose top depth because it is more straightforward to
validate against known data
  #       whereas the midpoints required some more thought. now that I am
pleased with stability of algo,
  #       I think it is worth switching to midpoint given it probably gives a
better estimate of
```

```
  #       the thickness of transitional zone between horizons when it matters
  #

  depthz <- c("hzdept","hzdepb")
  middepth <- hz[[depthz[1]]] + (hz[[depthz[2]]] - hz[[depthz[1]]]) / 2 #
TODO: evaluate

  increase.var <- hz[[attr]]

  threshold.vector <- threshold.fun(increase.var)

  if(length(threshold.vector) == 1) {
    # if threshold.fun() returns a constant value, expand it to match the
length of the attribute
    threshold.vector <- rep(threshold.vector, length(increase.var))

  } else if(length(threshold.vector) != length(increase.var)) {
    # this function assumes that the threshold.fun() supplied by the user
returns either a constant,
    # or a vector of equal length to `increase.var` when called above.
otherwise, we cannot calculate the result.
    # note that the threshold.fun() result _is_ allowed to contain NA, but
that will result in no output for affected cells
    stop("threshold.fun() result should be length 1 or equal to length of
attribute \'",attr,"\' (n=",length(increase.var),").")
  }

  # repeat attr content by horizon in columns; number of columns = number of
horizons
  attr.mat <- outer(increase.var, rep(1, length(increase.var)))

  # create a matrix of corresponding thresholds
  thresh.mat <- outer(threshold.vector, rep(1, length(increase.var)))

  # calculate a matrix of attr differences (between all horizons, not just
adjacent)
  attr.inc.mat <- outer(increase.var, increase.var, `-`)

  # calculate a vertical distance matrix (between all horizons)
  vdist.mat <- outer(middepth, c(0, middepth[-length(middepth)]), `-`)

  # crit1 "an increase of at least [thresh.mat]"
  increase.met <- (attr.mat - thresh.mat) > (attr.inc.mat *
upper.tri(attr.inc.mat))
  increase.met[is.na(increase.met)] <- FALSE

  # crit2 "within a vertical distance of [vertical.distance]"
  vdist.met <- abs(vdist.mat) <= vertical.distance

  # are crit 1 and crit 2 met?
  criteria.met <- increase.met & vdist.met
  return(criteria.met)
}
```

```
get.increase.matrix.oxic <- function(hz, attr, threshold.fun,
vertical.distance) {
  # p - a SoilProfileCollection, containing a single profile
  # attr - attribute name to get the "increase" of
  # threshold.fun - a function that returns the threshold (as a function of
attr); may return a constant single value
  # vertical distance - the vertical distance (determined from difference SPC
top depth variable) within which increase must be met

  # NOTE: should topdepths or midpoints be used? typically the same horizon
index is chosen
  #
  #       with thick horizons/diffuse transitions -- the midpoint is probably
more appropriate.
  #
  #       I originally chose top depth because it is more straightforward to
validate against known data
  #       whereas the midpoints required some more thought. now that I am
pleased with stability of algo,
  #       I think it is worth switching to midpoint given it probably gives a
better estimate of
  #       the thickness of transitional zone between horizons when it matters
  #

  depthz <- c("hzdept","hzdepb")
  middepth <- hz[[depthz[1]]] + (hz[[depthz[2]]] - hz[[depthz[1]]]) / 2 #
TODO: evaluate

  increase.var <- hz[[attr]]

  threshold.vector <- threshold.fun(increase.var)

  if(length(threshold.vector) == 1) {
    # if threshold.fun() returns a constant value, expand it to match the
length of the attribute
    threshold.vector <- rep(threshold.vector, length(increase.var))

  } else if(length(threshold.vector) != length(increase.var)) {
    # this function assumes that the threshold.fun() supplied by the user
returns either a constant,
    # or a vector of equal length to `increase.var` when called above.
otherwise, we cannot calculate the result.
    # note that the threshold.fun() result _is_ allowed to contain NA, but
that will result in no output for affected cells
    stop("threshold.fun() result should be length 1 or equal to length of
attribute \'",attr,"\' (n=",length(increase.var),").")
  }

  # repeat attr content by horizon in columns; number of columns = number of
horizons
  attr.mat <- outer(increase.var, rep(1, length(increase.var)))

  # create a matrix of corresponding thresholds
  thresh.mat <- outer(threshold.vector, rep(1, length(increase.var)))
```

```
  # calculate a matrix of attr differences (between all horizons, not just
adjacent)
  attr.inc.mat <- outer(increase.var, increase.var, `-`)

  # calculate a vertical distance matrix (between all horizons)
  vdist.mat <- outer(middepth, c(0, middepth[-length(middepth)]), `-`)

  # crit1 "an increase of at least [thresh.mat]"
  increase.met <- (attr.mat - thresh.mat) < (attr.inc.mat *
upper.tri(attr.inc.mat))
  increase.met[is.na(increase.met)] <- FALSE

  # crit2 "within a vertical distance of [vertical.distance]"
  vdist.met <- abs(vdist.mat) <= vertical.distance

  # are crit 1 and crit 2 met?
  criteria.met <- increase.met & vdist.met
  return(criteria.met)
}

get.increase.depths <- function(hz, attr, threshold.fun, vertical.distance) {
  topdepth <- hz$hzdept
  #TODO: is there interesting pedogenic information that can be derived
  # from the lower triangle of the criteria.met matrix we omit from the
  # "increase" matrix?) -- i.e. pale/haplo clay decrease at depth?
  criteria.met <- get.increase.matrix(hz, attr, threshold.fun,
vertical.distance)
  # get the index of  _first_ column from matrix criteria.met that has nonzero
sum,
  # this column index is the index of horizon where the attr increase is met
  # then return the top depth.
  return(topdepth[which(colSums(criteria.met) > 0)])
}

get.increase.depths.oxic <- function(hz, attr, threshold.fun,
vertical.distance) {
  topdepth <- hz$hzdept
  #TODO: is there interesting pedogenic information that can be derived
  # from the lower triangle of the criteria.met matrix we omit from the
  # "increase" matrix?) -- i.e. pale/haplo clay decrease at depth?
  criteria.met <- get.increase.matrix.oxic(hz, attr, threshold.fun,
vertical.distance)
  # get the index of  _first_ column from matrix criteria.met that has nonzero
sum,
  # this column index is the index of horizon where the attr increase is met
  # then return the top depth.
  return(topdepth[which(colSums(criteria.met) > 0)])
}



argillic.clay.increase.depth <- function(hz, clay.attr = 'claytotal') {
  vd <- 30
```

```
  return(get.increase.depths(hz, attr = clay.attr,
                             threshold.fun = crit.clay.argillic,
                             vertical.distance = vd)[1])
}

kandic.clay.increase.depth <- function(hz, clay.attr = 'claytotal') {
  vd <- 15
  return(get.increase.depths(hz, attr = clay.attr,
                             threshold.fun = crit.clay.kandic,
                             vertical.distance = vd)[1])
}

oxic.clay.increase.depth <- function(hz, clay.attr = 'claytotal') {
  vd <- 15
  return(get.increase.depths.oxic(hz, attr = clay.attr,
                                  threshold.fun = crit.clay.oxic,
                                  vertical.distance = vd)[1])
}

glom_br <- function(hz, pedon_id, top, bottom, truncate=FALSE) {
  #' Adapted glom function for base R to apply to these function.
  #' The original glom seems overly complicated for our purposes.
  #' We just need to return the rows within the depth range.

  hz <- dplyr::filter(hz, pedon_id==pedon_id)

  if(truncate) {

    g <- prof_slice(bottom=bottom,top=top,hz=hz)

    return(g)

  }

  g_top <- min(which(between(hz$hzdept,top,bottom)))
  g_bot <- max(which(between(hz$hzdepb,top,bottom)))
  g_idx <- c(g_top:g_bot)

  g <- hz[g_idx,]

  return(g)

}
```

Appendix O:

ST_logic.rda

order	suborder	greatgroup	subgroup	order_id	suborder_id	gg_id	subgroup_id	id
gelisols	histels	folistels	lithic folistels	A	A	A	A	AAAA
gelisols	histels	folistels	glacic folistels	A	A	A	B	AAAB
gelisols	histels	folistels	typic folistels	A	A	A	C	AAAC
gelisols	histels	glacistels	hemic glacistels	A	A	B	A	AABA
gelisols	histels	glacistels	sapric glacistels	A	A	B	B	AABB
gelisols	histels	glacistels	typic glacistels	A	A	B	C	AABC
gelisols	histels	fibristels	lithic fibristels	A	A	C	A	AACA
gelisols	histels	fibristels	terric fibristels	A	A	C	B	AACB
gelisols	histels	fibristels	fluvaquentic fibristels	A	A	C	C	AACC
gelisols	histels	fibristels	sphagnic fibristels	A	A	C	D	AACD
gelisols	histels	fibristels	typic fibristels	A	A	C	E	AACE
gelisols	histels	hemistels	lithic hemistels	A	A	D	A	AADA
gelisols	histels	hemistels	terric hemistels	A	A	D	B	AADB
gelisols	histels	hemistels	fluvaquentic hemistels	A	A	D	C	AADC
gelisols	histels	hemistels	typic hemistels	A	A	D	D	AADD
gelisols	histels	sapristels	lithic sapristels	A	A	E	A	AAEA
gelisols	histels	sapristels	terric sapristels	A	A	E	B	AAEB
gelisols	histels	sapristels	fluvaquentic sapristels	A	A	E	C	AAEC
gelisols	histels	sapristels	typic sapristels	A	A	E	D	AAED
gelisols	turbels	histoturbels	lithic histoturbels	A	B	A	A	ABAA
gelisols	turbels	histoturbels	glacic histoturbels	A	B	A	B	ABAB
gelisols	turbels	histoturbels	ruptic histoturbels	A	B	A	C	ABAC
gelisols	turbels	histoturbels	typic histoturbels	A	B	A	D	ABAD
gelisols	turbels	aquiturbels	lithic aquiturbels	A	B	B	A	ABBA
gelisols	turbels	aquiturbels	glacic aquiturbels	A	B	B	B	ABBB
gelisols	turbels	aquiturbels	sulfuric aquiturbels	A	B	B	C	ABBC
gelisols	turbels	aquiturbels	ruptic-histic aquiturbels	A	B	B	D	ABBD
gelisols	turbels	aquiturbels	psammentic aquiturbels	A	B	B	E	ABBE
gelisols	turbels	aquiturbels	typic aquiturbels	A	B	B	F	ABBF
gelisols	turbels	anhyturbels	lithic anhyturbels	A	B	C	A	ABCA
gelisols	turbels	anhyturbels	glacic anhyturbels	A	B	C	B	ABCB
gelisols	turbels	anhyturbels	petrogypsic anhyturbels	A	B	C	C	ABCC
gelisols	turbels	anhyturbels	gypsic anhyturbels	A	B	C	D	ABCD
gelisols	turbels	anhyturbels	nitric anhyturbels	A	B	C	E	ABCE
gelisols	turbels	anhyturbels	salic anhyturbels	A	B	C	F	ABCF
gelisols	turbels	anhyturbels	calcic anhyturbels	A	B	C	G	ABCG
gelisols	turbels	anhyturbels	typic anhyturbels	A	B	C	H	ABCH
gelisols	turbels	molliturbels	lithic molliturbels	A	B	D	A	ABDA
gelisols	turbels	molliturbels	glacic molliturbels	A	B	D	B	ABDB
gelisols	turbels	molliturbels	vertic molliturbels	A	B	D	C	ABDC
gelisols	turbels	molliturbels	andic molliturbels	A	B	D	D	ABDD
gelisols	turbels	molliturbels	vitrandic molliturbels	A	B	D	E	ABDE
gelisols	turbels	molliturbels	folistic molliturbels	A	B	D	F	ABDF
gelisols	turbels	molliturbels	cumulic molliturbels	A	B	D	G	ABDG
gelisols	turbels	molliturbels	aquic molliturbels	A	B	D	H	ABDH
gelisols	turbels	molliturbels	typic molliturbels	A	B	D	I	ABDI
gelisols	turbels	umbriturbels	lithic umbriturbels	A	B	E	A	ABEA
gelisols	turbels	umbriturbels	glacic umbriturbels	A	B	E	B	ABEB
gelisols	turbels	umbriturbels	vertic umbriturbels	A	B	E	C	ABEC
gelisols	turbels	umbriturbels	andic umbriturbels	A	B	E	D	ABED
gelisols	turbels	umbriturbels	vitrandic umbriturbels	A	B	E	E	ABEE
gelisols	turbels	umbriturbels	folistic umbriturbels	A	B	E	F	ABEF
gelisols	turbels	umbriturbels	cumulic umbriturbels	A	B	E	G	ABEG
gelisols	turbels	umbriturbels	aquic umbriturbels	A	B	E	H	ABEH
gelisols	turbels	umbriturbels	typic umbriturbels	A	B	E	I	ABEI
gelisols	turbels	psammoturbels	lithic psammoturbels	A	B	F	A	ABFA
gelisols	turbels	psammoturbels	glacic psammoturbels	A	B	F	B	ABFB
gelisols	turbels	psammoturbels	spodic psammoturbels	A	B	F	C	ABFC
gelisols	turbels	psammoturbels	typic psammoturbels	A	B	F	D	ABFD

gelisols	turbels	haploturbels	lithic haploturbels	A	B	G	A	ABGA
gelisols	turbels	haploturbels	glacic haploturbels	A	B	G	B	ABGB
gelisols	turbels	haploturbels	folistic haploturbels	A	B	G	C	ABGC
gelisols	turbels	haploturbels	aquic haploturbels	A	B	G	D	ABGD
gelisols	turbels	haploturbels	typic haploturbels	A	B	G	E	ABGE
gelisols	orthels	historthels	lithic historthels	A	C	A	A	ACAA
gelisols	orthels	historthels	glacic historthels	A	C	A	B	ACAB
gelisols	orthels	historthels	fluvaquentic historthels	A	C	A	C	ACAC
gelisols	orthels	historthels	fluventic historthels	A	C	A	D	ACAD
gelisols	orthels	historthels	ruptic historthels	A	C	A	E	ACAE
gelisols	orthels	historthels	typic historthels	A	C	A	F	ACAF
gelisols	orthels	aquorthels	lithic aquorthels	A	C	B	A	ACBA
gelisols	orthels	aquorthels	glacic aquorthels	A	C	B	B	ACBB
gelisols	orthels	aquorthels	sulfuric aquorthels	A	C	B	C	ACBC
gelisols	orthels	aquorthels	ruptic-histic aquorthels	A	C	B	D	ACBD
gelisols	orthels	aquorthels	andic aquorthels	A	C	B	E	ACBE
gelisols	orthels	aquorthels	vitrandic aquorthels	A	C	B	F	ACBF
gelisols	orthels	aquorthels	salic aquorthels	A	C	B	G	ACBG
gelisols	orthels	aquorthels	psammentic aquorthels	A	C	B	H	ACBH
gelisols	orthels	aquorthels	fluvaquentic aquorthels	A	C	B	I	ACBI
gelisols	orthels	aquorthels	typic aquorthels	A	C	B	J	ACBJ
gelisols	orthels	anhyorthels	lithic anhyorthels	A	C	C	A	ACCA
gelisols	orthels	anhyorthels	glacic anhyorthels	A	C	C	B	ACCB
gelisols	orthels	anhyorthels	petrogypsic anhyorthels	A	C	C	C	ACCC
gelisols	orthels	anhyorthels	gypsic anhyorthels	A	C	C	D	ACCD
gelisols	orthels	anhyorthels	nitric anhyorthels	A	C	C	E	ACCE
gelisols	orthels	anhyorthels	salic anhyorthels	A	C	C	F	ACCF
gelisols	orthels	anhyorthels	calcic anhyorthels	A	C	C	G	ACCG
gelisols	orthels	anhyorthels	typic anhyorthels	A	C	C	H	ACCH
gelisols	orthels	mollorthels	lithic mollorthels	A	C	D	A	ACDA
gelisols	orthels	mollorthels	glacic mollorthels	A	C	D	B	ACDB
gelisols	orthels	mollorthels	vertic mollorthels	A	C	D	C	ACDC
gelisols	orthels	mollorthels	andic mollorthels	A	C	D	D	ACDD
gelisols	orthels	mollorthels	vitrandic mollorthels	A	C	D	E	ACDE
gelisols	orthels	mollorthels	folistic mollorthels	A	C	D	F	ACDF
gelisols	orthels	mollorthels	cumulic mollorthels	A	C	D	G	ACDG
gelisols	orthels	mollorthels	aquic mollorthels	A	C	D	H	ACDH
gelisols	orthels	mollorthels	typic mollorthels	A	C	D	I	ACDI
gelisols	orthels	umbrorthels	lithic umbrorthels	A	C	E	A	ACEA
gelisols	orthels	umbrorthels	glacic umbrorthels	A	C	E	B	ACEB
gelisols	orthels	umbrorthels	vertic umbrorthels	A	C	E	C	ACEC
gelisols	orthels	umbrorthels	andic umbrorthels	A	C	E	D	ACED
gelisols	orthels	umbrorthels	vitrandic umbrorthels	A	C	E	E	ACEE
gelisols	orthels	umbrorthels	folistic umbrorthels	A	C	E	F	ACEF
gelisols	orthels	umbrorthels	cumulic umbrorthels	A	C	E	G	ACEG
gelisols	orthels	umbrorthels	aquic umbrorthels	A	C	E	H	ACEH
gelisols	orthels	umbrorthels	typic umbrorthels	A	C	E	I	ACEI
gelisols	orthels	argiorthels	lithic argiorthels	A	C	F	A	ACFA
gelisols	orthels	argiorthels	glacic argiorthels	A	C	F	B	ACFB
gelisols	orthels	argiorthels	natric argiorthels	A	C	F	C	ACFC
gelisols	orthels	argiorthels	typic argiorthels	A	C	F	D	ACFD
gelisols	orthels	psammorthels	lithic psammorthels	A	C	G	A	ACGA
gelisols	orthels	psammorthels	glacic psammorthels	A	C	G	B	ACGB
gelisols	orthels	psammorthels	spodic psammorthels	A	C	G	C	ACGC
gelisols	orthels	psammorthels	typic psammorthels	A	C	G	D	ACGD
gelisols	orthels	haplorthels	lithic haplorthels	A	C	H	A	ACHA
gelisols	orthels	haplorthels	glacic haplorthels	A	C	H	B	ACHB
gelisols	orthels	haplorthels	fluvaquentic haplorthels	A	C	H	C	ACHC
gelisols	orthels	haplorthels	folistic haplorthels	A	C	H	D	ACHD
gelisols	orthels	haplorthels	aquic haplorthels	A	C	H	E	ACHE

gelisols	orthels	haplorthels	fluventic haplorthels	A	C	H	F	ACHF
gelisols	orthels	haplorthels	typic haplorthels	A	C	H	G	ACHG
histosols	folists	cryofolists	lithic cryofolists	B	A	A	A	BAAA
histosols	folists	cryofolists	typic cryofolists	B	A	A	B	BAAB
histosols	folists	torrifolists	lithic torrifolists	B	A	B	A	BABA
histosols	folists	torrifolists	typic torrifolists	B	A	B	B	BABB
histosols	folists	ustifolists	lithic ustifolists	B	A	C	A	BACA
histosols	folists	ustifolists	typic ustifolists	B	A	C	B	BACB
histosols	folists	udifolists	lithic udifolists	B	A	D	A	BADA
histosols	folists	udifolists	typic udifolists	B	A	D	B	BADB
histosols	wassists	frasiwassists	fibric frasiwassists	B	B	A	A	BBAA
histosols	wassists	frasiwassists	sapric frasiwassists	B	B	A	B	BBAB
histosols	wassists	frasiwassists	typic frasiwassists	B	B	A	C	BBAC
histosols	wassists	sulfiwassists	fibric sulfiwassists	B	B	B	A	BBBA
histosols	wassists	sulfiwassists	sapric sulfiwassists	B	B	B	B	BBBB
histosols	wassists	sulfiwassists	typic sulfiwassists	B	B	B	C	BBBC
histosols	wassists	haplowassists	sulfic haplowassists	B	B	C	A	BBCA
histosols	wassists	haplowassists	fibric haplowassists	B	B	C	B	BBCB
histosols	wassists	haplowassists	sapric haplowassists	B	B	C	C	BBCC
histosols	wassists	haplowassists	typic haplowassists	B	B	C	D	BBCD
histosols	fibrists	cryofibrists	hydric cryofibrists	B	C	A	A	BCAA
histosols	fibrists	cryofibrists	lithic cryofibrists	B	C	A	B	BCAB
histosols	fibrists	cryofibrists	terric cryofibrists	B	C	A	C	BCAC
histosols	fibrists	cryofibrists	fluvaquentic cryofibrists	B	C	A	D	BCAD
histosols	fibrists	cryofibrists	sphagnic cryofibrists	B	C	A	E	BCAE
histosols	fibrists	cryofibrists	typic cryofibrists	B	C	A	F	BCAF
histosols	fibrists	sphagnofibrists	hydric sphagnofibrists	B	C	B	A	BCBA
histosols	fibrists	sphagnofibrists	lithic sphagnofibrists	B	C	B	B	BCBB
histosols	fibrists	sphagnofibrists	limnic sphagnofibrists	B	C	B	C	BCBC
histosols	fibrists	sphagnofibrists	terric sphagnofibrists	B	C	B	D	BCBD
histosols	fibrists	sphagnofibrists	fluvaquentic sphagnofibrists	B	C	B	E	BCBE
histosols	fibrists	sphagnofibrists	hemic sphagnofibrists	B	C	B	F	BCBF
histosols	fibrists	sphagnofibrists	typic sphagnofibrists	B	C	B	G	BCBG
histosols	fibrists	haplofibrists	hydric haplofibrists	B	C	C	A	BCCA
histosols	fibrists	haplofibrists	lithic haplofibrists	B	C	C	B	BCCB
histosols	fibrists	haplofibrists	limnic haplofibrists	B	C	C	C	BCCC
histosols	fibrists	haplofibrists	terric haplofibrists	B	C	C	D	BCCD
histosols	fibrists	haplofibrists	fluvaquentic haplofibrists	B	C	C	E	BCCE
histosols	fibrists	haplofibrists	hemic haplofibrists	B	C	C	F	BCCF
histosols	fibrists	haplofibrists	typic haplofibrists	B	C	C	G	BCCG
histosols	saprists	sulfosaprists	typic sulfosaprists	B	D	A	A	BDAA
histosols	saprists	sulfisaprists	terric sulfisaprists	B	D	B	A	BDBA
histosols	saprists	sulfisaprists	typic sulfisaprists	B	D	B	B	BDBB
histosols	saprists	cryosaprists	lithic cryosaprists	B	D	C	A	BDCA
histosols	saprists	cryosaprists	limnic cryosaprists	B	D	C	B	BDCB
histosols	saprists	cryosaprists	terric cryosaprists	B	D	C	C	BDCC
histosols	saprists	cryosaprists	fluvaquentic cryosaprists	B	D	C	D	BDCD
histosols	saprists	cryosaprists	typic cryosaprists	B	D	C	E	BDCE
histosols	saprists	haplosaprists	lithic haplosaprists	B	D	D	A	BDDA
histosols	saprists	haplosaprists	limnic haplosaprists	B	D	D	B	BDDB
histosols	saprists	haplosaprists	halic terric haplosaprists	B	D	D	C	BDDC
histosols	saprists	haplosaprists	halic haplosaprists	B	D	D	D	BDDD
histosols	saprists	haplosaprists	terric haplosaprists	B	D	D	E	BDDE
histosols	saprists	haplosaprists	fluvaquentic haplosaprists	B	D	D	F	BDDF
histosols	saprists	haplosaprists	hemic haplosaprists	B	D	D	G	BDDG
histosols	saprists	haplosaprists	typic haplosaprists	B	D	D	H	BDDH
histosols	hemists	sulfohemists	typic sulfohemists	B	E	A	A	BEAA
histosols	hemists	sulfihemists	terric sulfihemists	B	E	B	A	BEBA
histosols	hemists	sulfihemists	typic sulfihemists	B	E	B	B	BEBB
histosols	hemists	luvihemists	typic luvihemists	B	E	C	A	BECA

histosols	hemists	cryohemists	hydric cryohemists	B	E	D	A	BEDA
histosols	hemists	cryohemists	lithic cryohemists	B	E	D	B	BEDB
histosols	hemists	cryohemists	terric cryohemists	B	E	D	C	BEDC
histosols	hemists	cryohemists	fluvaquentic cryohemists	B	E	D	D	BEDD
histosols	hemists	cryohemists	typic cryohemists	B	E	D	E	BEDE
histosols	hemists	haplohemists	hydric haplohemists	B	E	E	A	BEEA
histosols	hemists	haplohemists	lithic haplohemists	B	E	E	B	BEEB
histosols	hemists	haplohemists	limnic haplohemists	B	E	E	C	BEEC
histosols	hemists	haplohemists	terric haplohemists	B	E	E	D	BEED
histosols	hemists	haplohemists	fluvaquentic haplohemists	B	E	E	E	BEEE
histosols	hemists	haplohemists	fibric haplohemists	B	E	E	F	BEEF
histosols	hemists	haplohemists	sapric haplohemists	B	E	E	G	BEEG
histosols	hemists	haplohemists	typic haplohemists	B	E	E	H	BEEH
spodosols	aquods	cryaquods	lithic cryaquods	C	A	A	A	CAAA
spodosols	aquods	cryaquods	placic cryaquods	C	A	A	B	CAAB
spodosols	aquods	cryaquods	duric cryaquods	C	A	A	C	CAAC
spodosols	aquods	cryaquods	andic cryaquods	C	A	A	D	CAAD
spodosols	aquods	cryaquods	entic cryaquods	C	A	A	E	CAAE
spodosols	aquods	cryaquods	typic cryaquods	C	A	A	F	CAAF
spodosols	aquods	alaquods	lithic alaquods	C	A	B	A	CABA
spodosols	aquods	alaquods	duric alaquods	C	A	B	B	CABB
spodosols	aquods	alaquods	histic alaquods	C	A	B	C	CABC
spodosols	aquods	alaquods	alfic arenic alaquods	C	A	B	D	CABD
spodosols	aquods	alaquods	arenic ultic alaquods	C	A	B	E	CABE
spodosols	aquods	alaquods	arenic umbric alaquods	C	A	B	F	CABF
spodosols	aquods	alaquods	arenic alaquods	C	A	B	G	CABG
spodosols	aquods	alaquods	grossarenic alaquods	C	A	B	H	CABH
spodosols	aquods	alaquods	alfic alaquods	C	A	B	I	CABI
spodosols	aquods	alaquods	ultic alaquods	C	A	B	J	CABJ
spodosols	aquods	alaquods	aeric alaquods	C	A	B	K	CABK
spodosols	aquods	alaquods	typic alaquods	C	A	B	L	CABL
spodosols	aquods	fragiaquods	histic fragiaquods	C	A	C	A	CACA
spodosols	aquods	fragiaquods	haploplaggic fragiaquods	C	A	C	B	CACB
spodosols	aquods	fragiaquods	argic fragiaquods	C	A	C	C	CACC
spodosols	aquods	fragiaquods	typic fragiaquods	C	A	C	D	CACD
spodosols	aquods	placaquods	andic placaquods	C	A	D	A	CADA
spodosols	aquods	placaquods	typic placaquods	C	A	D	B	CADB
spodosols	aquods	duraquods	histic duraquods	C	A	E	A	CAEA
spodosols	aquods	duraquods	andic duraquods	C	A	E	B	CAEB
spodosols	aquods	duraquods	typic duraquods	C	A	E	C	CAEC
spodosols	aquods	epiaquods	lithic epiaquods	C	A	F	A	CAFA
spodosols	aquods	epiaquods	histic epiaquods	C	A	F	B	CAFB
spodosols	aquods	epiaquods	andic epiaquods	C	A	F	C	CAFC
spodosols	aquods	epiaquods	alfic epiaquods	C	A	F	D	CAFD
spodosols	aquods	epiaquods	ultic epiaquods	C	A	F	E	CAFE
spodosols	aquods	epiaquods	umbric epiaquods	C	A	F	F	CAFF
spodosols	aquods	epiaquods	typic epiaquods	C	A	F	G	CAFG
spodosols	aquods	endoaquods	lithic endoaquods	C	A	G	A	CAGA
spodosols	aquods	endoaquods	histic endoaquods	C	A	G	B	CAGB
spodosols	aquods	endoaquods	andic endoaquods	C	A	G	C	CAGC
spodosols	aquods	endoaquods	argic endoaquods	C	A	G	D	CAGD
spodosols	aquods	endoaquods	umbric endoaquods	C	A	G	E	CAGE
spodosols	aquods	endoaquods	typic endoaquods	C	A	G	F	CAGF
spodosols	gelods	humigelods	lithic humigelods	C	B	A	A	CBAA
spodosols	gelods	humigelods	andic humigelods	C	B	A	B	CBAB
spodosols	gelods	humigelods	aquic humigelods	C	B	A	C	CBAC
spodosols	gelods	humigelods	turbic humigelods	C	B	A	D	CBAD
spodosols	gelods	humigelods	typic humigelods	C	B	A	E	CBAE
spodosols	gelods	haplogelods	lithic haplogelods	C	B	B	A	CBBA
spodosols	gelods	haplogelods	andic haplogelods	C	B	B	B	CBBB

spodosols	gelods	haplogelods	aquic haplogelods	C	B	B	C	CBBC
spodosols	gelods	haplogelods	turbic haplogelods	C	B	B	D	CBBD
spodosols	gelods	haplogelods	typic haplogelods	C	B	B	E	CBBE
spodosols	cryods	placocryods	andic placocryods	C	C	A	A	CCAA
spodosols	cryods	placocryods	humic placocryods	C	C	A	B	CCAB
spodosols	cryods	placocryods	typic placocryods	C	C	A	C	CCAC
spodosols	cryods	duricryods	aquandic duricryods	C	C	B	A	CCBA
spodosols	cryods	duricryods	andic duricryods	C	C	B	B	CCBB
spodosols	cryods	duricryods	aquic duricryods	C	C	B	C	CCBC
spodosols	cryods	duricryods	oxyaquic duricryods	C	C	B	D	CCBD
spodosols	cryods	duricryods	humic duricryods	C	C	B	E	CCBE
spodosols	cryods	duricryods	typic duricryods	C	C	B	F	CCBF
spodosols	cryods	humicryods	lithic humicryods	C	C	C	A	CCCA
spodosols	cryods	humicryods	aquandic humicryods	C	C	C	B	CCCB
spodosols	cryods	humicryods	andic humicryods	C	C	C	C	CCCC
spodosols	cryods	humicryods	folistic humicryods	C	C	C	D	CCCD
spodosols	cryods	humicryods	aquic humicryods	C	C	C	E	CCCE
spodosols	cryods	humicryods	oxyaquic humicryods	C	C	C	F	CCCF
spodosols	cryods	humicryods	typic humicryods	C	C	C	G	CCCG
spodosols	cryods	haplocryods	lithic haplocryods	C	C	D	A	CCDA
spodosols	cryods	haplocryods	aquandic haplocryods	C	C	D	B	CCDB
spodosols	cryods	haplocryods	andic haplocryods	C	C	D	C	CCDC
spodosols	cryods	haplocryods	folistic haplocryods	C	C	D	D	CCDD
spodosols	cryods	haplocryods	aquic haplocryods	C	C	D	E	CCDE
spodosols	cryods	haplocryods	oxyaquic haplocryods	C	C	D	F	CCDF
spodosols	cryods	haplocryods	entic haplocryods	C	C	D	G	CCDG
spodosols	cryods	haplocryods	typic haplocryods	C	C	D	H	CCDH
spodosols	humods	placohumods	andic placohumods	C	D	A	A	CDAA
spodosols	humods	placohumods	typic placohumods	C	D	A	B	CDAB
spodosols	humods	durihumods	andic durihumods	C	D	B	A	CDBA
spodosols	humods	durihumods	typic durihumods	C	D	B	B	CDBB
spodosols	humods	fragihumods	typic fragihumods	C	D	C	A	CDCA
spodosols	humods	haplohumods	lithic haplohumods	C	D	D	A	CDDA
spodosols	humods	haplohumods	andic haplohumods	C	D	D	B	CDDB
spodosols	humods	haplohumods	haploplaggic haplohumods	C	D	D	C	CDDC
spodosols	humods	haplohumods	typic haplohumods	C	D	D	D	CDDD
spodosols	orthods	placorthods	typic placorthods	C	E	A	A	CEAA
spodosols	orthods	durorthods	andic durorthods	C	E	B	A	CEBA
spodosols	orthods	durorthods	typic durorthods	C	E	B	B	CEBB
spodosols	orthods	fragiorthods	aquic fragiorthods	C	E	C	A	CECA
spodosols	orthods	fragiorthods	alfic oxyaquic fragiorthods	C	E	C	B	CECB
spodosols	orthods	fragiorthods	oxyaquic fragiorthods	C	E	C	C	CECC
spodosols	orthods	fragiorthods	haploplaggic fragiorthods	C	E	C	D	CECD
spodosols	orthods	fragiorthods	alfic fragiorthods	C	E	C	E	CECE
spodosols	orthods	fragiorthods	ultic fragiorthods	C	E	C	F	CECF
spodosols	orthods	fragiorthods	entic fragiorthods	C	E	C	G	CECG
spodosols	orthods	fragiorthods	typic fragiorthods	C	E	C	H	CECH
spodosols	orthods	alorthods	oxyaquic alorthods	C	E	D	A	CEDA
spodosols	orthods	alorthods	arenic ultic alorthods	C	E	D	B	CEDB
spodosols	orthods	alorthods	arenic alorthods	C	E	D	C	CEDC
spodosols	orthods	alorthods	entic grossarenic alorthods	C	E	D	D	CEDD
spodosols	orthods	alorthods	entic alorthods	C	E	D	E	CEDE
spodosols	orthods	alorthods	grossarenic alorthods	C	E	D	F	CEDF
spodosols	orthods	alorthods	haploplaggic alorthods	C	E	D	G	CEDG
spodosols	orthods	alorthods	alfic alorthods	C	E	D	H	CEDH
spodosols	orthods	alorthods	ultic alorthods	C	E	D	I	CEDI
spodosols	orthods	alorthods	typic alorthods	C	E	D	J	CEDJ
spodosols	orthods	haplorthods	entic lithic haplorthods	C	E	E	A	CEEA
spodosols	orthods	haplorthods	lithic haplorthods	C	E	E	B	CEEB
spodosols	orthods	haplorthods	fragiaquic haplorthods	C	E	E	C	CEEC

spodosols	orthods	haplorthods	aqualfic haplorthods	C	E	E	D	CEED
spodosols	orthods	haplorthods	aquentic haplorthods	C	E	E	E	CEEE
spodosols	orthods	haplorthods	aquic haplorthods	C	E	E	F	CEEF
spodosols	orthods	haplorthods	alfic oxyaquic haplorthods	C	E	E	G	CEEG
spodosols	orthods	haplorthods	oxyaquic ultic haplorthods	C	E	E	H	CEEH
spodosols	orthods	haplorthods	fragic haplorthods	C	E	E	I	CEEI
spodosols	orthods	haplorthods	lamellic oxyaquic haplorthods	C	E	E	J	CEEJ
spodosols	orthods	haplorthods	lamellic haplorthods	C	E	E	K	CEEK
spodosols	orthods	haplorthods	oxyaquic haplorthods	C	E	E	L	CEEL
spodosols	orthods	haplorthods	andic haplorthods	C	E	E	M	CEEM
spodosols	orthods	haplorthods	alfic haplorthods	C	E	E	N	CEEN
spodosols	orthods	haplorthods	ultic haplorthods	C	E	E	O	CEEO
spodosols	orthods	haplorthods	entic haplorthods	C	E	E	P	CEEP
spodosols	orthods	haplorthods	typic haplorthods	C	E	E	Q	CEEQ
andisols	aquands	gelaquands	histic gelaquands	D	A	A	A	DAAA
andisols	aquands	gelaquands	turbic gelaquands	D	A	A	B	DAAB
andisols	aquands	gelaquands	thaptic gelaquands	D	A	A	C	DAAC
andisols	aquands	gelaquands	typic gelaquands	D	A	A	D	DAAD
andisols	aquands	cryaquands	lithic cryaquands	D	A	B	A	DABA
andisols	aquands	cryaquands	histic cryaquands	D	A	B	B	DABB
andisols	aquands	cryaquands	thaptic cryaquands	D	A	B	C	DABC
andisols	aquands	cryaquands	typic cryaquands	D	A	B	D	DABD
andisols	aquands	placaquands	lithic placaquands	D	A	C	A	DACA
andisols	aquands	placaquands	duric histic placaquands	D	A	C	B	DACB
andisols	aquands	placaquands	duric placaquands	D	A	C	C	DACC
andisols	aquands	placaquands	histic placaquands	D	A	C	D	DACD
andisols	aquands	placaquands	thaptic placaquands	D	A	C	E	DACE
andisols	aquands	placaquands	typic placaquands	D	A	C	F	DACF
andisols	aquands	duraquands	histic duraquands	D	A	D	A	DADA
andisols	aquands	duraquands	acraquoxic duraquands	D	A	D	B	DADB
andisols	aquands	duraquands	thaptic duraquands	D	A	D	C	DADC
andisols	aquands	duraquands	typic duraquands	D	A	D	D	DADD
andisols	aquands	vitraquands	lithic vitraquands	D	A	E	A	DAEA
andisols	aquands	vitraquands	duric vitraquands	D	A	E	B	DAEB
andisols	aquands	vitraquands	histic vitraquands	D	A	E	C	DAEC
andisols	aquands	vitraquands	thaptic vitraquands	D	A	E	D	DAED
andisols	aquands	vitraquands	typic vitraquands	D	A	E	E	DAEE
andisols	aquands	melanaquands	lithic melanaquands	D	A	F	A	DAFA
andisols	aquands	melanaquands	acraquoxic melanaquands	D	A	F	B	DAFB
andisols	aquands	melanaquands	hydric pachic melanaquands	D	A	F	C	DAFC
andisols	aquands	melanaquands	hydric melanaquands	D	A	F	D	DAFD
andisols	aquands	melanaquands	pachic melanaquands	D	A	F	E	DAFE
andisols	aquands	melanaquands	thaptic melanaquands	D	A	F	F	DAFF
andisols	aquands	melanaquands	typic melanaquands	D	A	F	G	DAFG
andisols	aquands	epiaquands	duric epiaquands	D	A	G	A	DAGA
andisols	aquands	epiaquands	histic epiaquands	D	A	G	B	DAGB
andisols	aquands	epiaquands	alic epiaquands	D	A	G	C	DAGC
andisols	aquands	epiaquands	hydric epiaquands	D	A	G	D	DAGD
andisols	aquands	epiaquands	thaptic epiaquands	D	A	G	E	DAGE
andisols	aquands	epiaquands	typic epiaquands	D	A	G	F	DAGF
andisols	aquands	endoaquands	lithic endoaquands	D	A	H	A	DAHA
andisols	aquands	endoaquands	duric endoaquands	D	A	H	B	DAHB
andisols	aquands	endoaquands	histic endoaquands	D	A	H	C	DAHC
andisols	aquands	endoaquands	alic endoaquands	D	A	H	D	DAHD
andisols	aquands	endoaquands	hydric endoaquands	D	A	H	E	DAHE
andisols	aquands	endoaquands	thaptic endoaquands	D	A	H	F	DAHF
andisols	aquands	endoaquands	typic endoaquands	D	A	H	G	DAHG
andisols	gelands	vitrigelands	humic vitrigelands	D	B	A	A	DBAA
andisols	gelands	vitrigelands	turbic vitrigelands	D	B	A	B	DBAB
andisols	gelands	vitrigelands	typic vitrigelands	D	B	A	C	DBAC

andisols	cryands	duricryands	aquic duricryands	D	C	A	A	DCAA
andisols	cryands	duricryands	eutric oxyaquic duricryands	D	C	A	B	DCAB
andisols	cryands	duricryands	oxyaquic duricryands	D	C	A	C	DCAC
andisols	cryands	duricryands	eutric duricryands	D	C	A	D	DCAD
andisols	cryands	duricryands	typic duricryands	D	C	A	E	DCAE
andisols	cryands	hydrocryands	lithic hydrocryands	D	C	B	A	DCBA
andisols	cryands	hydrocryands	placic hydrocryands	D	C	B	B	DCBB
andisols	cryands	hydrocryands	aquic hydrocryands	D	C	B	C	DCBC
andisols	cryands	hydrocryands	thaptic hydrocryands	D	C	B	D	DCBD
andisols	cryands	hydrocryands	typic hydrocryands	D	C	B	E	DCBE
andisols	cryands	melanocryands	lithic melanocryands	D	C	C	A	DCCA
andisols	cryands	melanocryands	vitric melanocryands	D	C	C	B	DCCB
andisols	cryands	melanocryands	typic melanocryands	D	C	C	C	DCCC
andisols	cryands	fulvicryands	lithic fulvicryands	D	C	D	A	DCDA
andisols	cryands	fulvicryands	folistic fulvicryands	D	C	D	B	DCDB
andisols	cryands	fulvicryands	eutric pachic fulvicryands	D	C	D	C	DCDC
andisols	cryands	fulvicryands	eutric fulvicryands	D	C	D	D	DCDD
andisols	cryands	fulvicryands	pachic fulvicryands	D	C	D	E	DCDE
andisols	cryands	fulvicryands	vitric fulvicryands	D	C	D	F	DCDF
andisols	cryands	fulvicryands	typic fulvicryands	D	C	D	G	DCDG
andisols	cryands	vitricryands	lithic vitricryands	D	C	E	A	DCEA
andisols	cryands	vitricryands	folistic vitricryands	D	C	E	B	DCEB
andisols	cryands	vitricryands	aquic vitricryands	D	C	E	C	DCEC
andisols	cryands	vitricryands	oxyaquic vitricryands	D	C	E	D	DCED
andisols	cryands	vitricryands	spodic vitricryands	D	C	E	E	DCEE
andisols	cryands	vitricryands	thaptic vitricryands	D	C	E	F	DCEF
andisols	cryands	vitricryands	humic xeric vitricryands	D	C	E	G	DCEG
andisols	cryands	vitricryands	xeric vitricryands	D	C	E	H	DCEH
andisols	cryands	vitricryands	ultic vitricryands	D	C	E	I	DCEI
andisols	cryands	vitricryands	alfic vitricryands	D	C	E	J	DCEJ
andisols	cryands	vitricryands	humic vitricryands	D	C	E	K	DCEK
andisols	cryands	vitricryands	typic vitricryands	D	C	E	L	DCEL
andisols	cryands	haplocryands	lithic haplocryands	D	C	F	A	DCFA
andisols	cryands	haplocryands	folistic haplocryands	D	C	F	B	DCFB
andisols	cryands	haplocryands	aquic haplocryands	D	C	F	C	DCFC
andisols	cryands	haplocryands	oxyaquic haplocryands	D	C	F	D	DCFD
andisols	cryands	haplocryands	alic haplocryands	D	C	F	E	DCFE
andisols	cryands	haplocryands	spodic haplocryands	D	C	F	F	DCFF
andisols	cryands	haplocryands	acrudoxic haplocryands	D	C	F	G	DCFG
andisols	cryands	haplocryands	vitric haplocryands	D	C	F	H	DCFH
andisols	cryands	haplocryands	thaptic haplocryands	D	C	F	I	DCFI
andisols	cryands	haplocryands	xeric haplocryands	D	C	F	J	DCFJ
andisols	cryands	haplocryands	typic haplocryands	D	C	F	K	DCFK
andisols	torrands	duritorrands	petrocalcic duritorrands	D	D	A	A	DDAA
andisols	torrands	duritorrands	vitric duritorrands	D	D	A	B	DDAB
andisols	torrands	duritorrands	typic duritorrands	D	D	A	C	DDAC
andisols	torrands	vitritorrands	lithic vitritorrands	D	D	B	A	DDBA
andisols	torrands	vitritorrands	duric vitritorrands	D	D	B	B	DDBB
andisols	torrands	vitritorrands	aquic vitritorrands	D	D	B	C	DDBC
andisols	torrands	vitritorrands	calcic vitritorrands	D	D	B	D	DDBD
andisols	torrands	vitritorrands	typic vitritorrands	D	D	B	E	DDBE
andisols	torrands	haplotorrands	lithic haplotorrands	D	D	C	A	DDCA
andisols	torrands	haplotorrands	duric haplotorrands	D	D	C	B	DDCB
andisols	torrands	haplotorrands	calcic haplotorrands	D	D	C	C	DDCC
andisols	torrands	haplotorrands	typic haplotorrands	D	D	C	D	DDCD
andisols	xerands	vitrixerands	lithic vitrixerands	D	E	A	A	DEAA
andisols	xerands	vitrixerands	aquic vitrixerands	D	E	A	B	DEAB
andisols	xerands	vitrixerands	thaptic vitrixerands	D	E	A	C	DEAC
andisols	xerands	vitrixerands	alfic humic vitrixerands	D	E	A	D	DEAD
andisols	xerands	vitrixerands	ultic vitrixerands	D	E	A	E	DEAE

andisols	xerands	vitrixerands	alfic vitrixerands	D	E	A	F	DEAF
andisols	xerands	vitrixerands	humic vitrixerands	D	E	A	G	DEAG
andisols	xerands	vitrixerands	typic vitrixerands	D	E	A	H	DEAH
andisols	xerands	melanoxerands	pachic melanoxerands	D	E	B	A	DEBA
andisols	xerands	melanoxerands	typic melanoxerands	D	E	B	B	DEBB
andisols	xerands	haploxerands	lithic haploxerands	D	E	C	A	DECA
andisols	xerands	haploxerands	aquic haploxerands	D	E	C	B	DECB
andisols	xerands	haploxerands	thaptic haploxerands	D	E	C	C	DECC
andisols	xerands	haploxerands	calcic haploxerands	D	E	C	D	DECD
andisols	xerands	haploxerands	ultic haploxerands	D	E	C	E	DECE
andisols	xerands	haploxerands	alfic humic haploxerands	D	E	C	F	DECF
andisols	xerands	haploxerands	alfic haploxerands	D	E	C	G	DECG
andisols	xerands	haploxerands	humic haploxerands	D	E	C	H	DECH
andisols	xerands	haploxerands	typic haploxerands	D	E	C	I	DECI
andisols	vitrands	ustivitrands	lithic ustivitrands	D	F	A	A	DFAA
andisols	vitrands	ustivitrands	aquic ustivitrands	D	F	A	B	DFAB
andisols	vitrands	ustivitrands	thaptic ustivitrands	D	F	A	C	DFAC
andisols	vitrands	ustivitrands	calcic ustivitrands	D	F	A	D	DFAD
andisols	vitrands	ustivitrands	humic ustivitrands	D	F	A	E	DFAE
andisols	vitrands	ustivitrands	typic ustivitrands	D	F	A	F	DFAF
andisols	vitrands	udivitrands	lithic udivitrands	D	F	B	A	DFBA
andisols	vitrands	udivitrands	aquic udivitrands	D	F	B	B	DFBB
andisols	vitrands	udivitrands	oxyaquic udivitrands	D	F	B	C	DFBC
andisols	vitrands	udivitrands	thaptic udivitrands	D	F	B	D	DFBD
andisols	vitrands	udivitrands	ultic udivitrands	D	F	B	E	DFBE
andisols	vitrands	udivitrands	alfic udivitrands	D	F	B	F	DFBF
andisols	vitrands	udivitrands	humic udivitrands	D	F	B	G	DFBG
andisols	vitrands	udivitrands	typic udivitrands	D	F	B	H	DFBH
andisols	ustands	durustands	aquic durustands	D	G	A	A	DGAA
andisols	ustands	durustands	thaptic durustands	D	G	A	B	DGAB
andisols	ustands	durustands	humic durustands	D	G	A	C	DGAC
andisols	ustands	durustands	typic durustands	D	G	A	D	DGAD
andisols	ustands	haplustands	lithic haplustands	D	G	B	A	DGBA
andisols	ustands	haplustands	aquic haplustands	D	G	B	B	DGBB
andisols	ustands	haplustands	dystric vitric haplustands	D	G	B	C	DGBC
andisols	ustands	haplustands	vitric haplustands	D	G	B	D	DGBD
andisols	ustands	haplustands	pachic haplustands	D	G	B	E	DGBE
andisols	ustands	haplustands	thaptic haplustands	D	G	B	F	DGBF
andisols	ustands	haplustands	calcic haplustands	D	G	B	G	DGBG
andisols	ustands	haplustands	dystric haplustands	D	G	B	H	DGBH
andisols	ustands	haplustands	oxic haplustands	D	G	B	I	DGBI
andisols	ustands	haplustands	ultic haplustands	D	G	B	J	DGBJ
andisols	ustands	haplustands	alfic haplustands	D	G	B	K	DGBK
andisols	ustands	haplustands	humic haplustands	D	G	B	L	DGBL
andisols	ustands	haplustands	typic haplustands	D	G	B	M	DGBM
andisols	udands	placudands	lithic placudands	D	H	A	A	DHAA
andisols	udands	placudands	aquic placudands	D	H	A	B	DHAB
andisols	udands	placudands	acrudoxic placudands	D	H	A	C	DHAC
andisols	udands	placudands	hydric placudands	D	H	A	D	DHAD
andisols	udands	placudands	typic placudands	D	H	A	E	DHAE
andisols	udands	durudands	aquic durudands	D	H	B	A	DHBA
andisols	udands	durudands	eutric durudands	D	H	B	B	DHBB
andisols	udands	durudands	acrudoxic durudands	D	H	B	C	DHBC
andisols	udands	durudands	hydric durudands	D	H	B	D	DHBD
andisols	udands	durudands	pachic durudands	D	H	B	E	DHBE
andisols	udands	durudands	typic durudands	D	H	B	F	DHBF
andisols	udands	melanudands	lithic melanudands	D	H	C	A	DHCA
andisols	udands	melanudands	anthraquic melanudands	D	H	C	B	DHCB
andisols	udands	melanudands	aquic melanudands	D	H	C	C	DHCC
andisols	udands	melanudands	acrudoxic vitric melanudands	D	H	C	D	DHCD

andisols	udands	melanudands	acrudoxic hydric melanudands	D	H	C	E	DHCE
andisols	udands	melanudands	acrudoxic melanudands	D	H	C	F	DHCF
andisols	udands	melanudands	pachic vitric melanudands	D	H	C	G	DHCG
andisols	udands	melanudands	vitric melanudands	D	H	C	H	DHCH
andisols	udands	melanudands	hydric pachic melanudands	D	H	C	I	DHCI
andisols	udands	melanudands	pachic melanudands	D	H	C	J	DHCJ
andisols	udands	melanudands	hydric melanudands	D	H	C	K	DHCK
andisols	udands	melanudands	thaptic melanudands	D	H	C	L	DHCL
andisols	udands	melanudands	ultic melanudands	D	H	C	M	DHCM
andisols	udands	melanudands	eutric melanudands	D	H	C	N	DHCN
andisols	udands	melanudands	typic melanudands	D	H	C	O	DHCO
andisols	udands	hydrudands	lithic hydrudands	D	H	D	A	DHDA
andisols	udands	hydrudands	aquic hydrudands	D	H	D	B	DHDB
andisols	udands	hydrudands	acrudoxic thaptic hydrudands	D	H	D	C	DHDC
andisols	udands	hydrudands	acrudoxic hydrudands	D	H	D	D	DHDD
andisols	udands	hydrudands	thaptic hydrudands	D	H	D	E	DHDE
andisols	udands	hydrudands	eutric hydrudands	D	H	D	F	DHDF
andisols	udands	hydrudands	ultic hydrudands	D	H	D	G	DHDG
andisols	udands	hydrudands	typic hydrudands	D	H	D	H	DHDH
andisols	udands	fulvudands	eutric lithic fulvudands	D	H	E	A	DHEA
andisols	udands	fulvudands	lithic fulvudands	D	H	E	B	DHEB
andisols	udands	fulvudands	aquic fulvudands	D	H	E	C	DHEC
andisols	udands	fulvudands	oxyaquic fulvudands	D	H	E	D	DHED
andisols	udands	fulvudands	hydric fulvudands	D	H	E	E	DHEE
andisols	udands	fulvudands	acrudoxic fulvudands	D	H	E	F	DHEF
andisols	udands	fulvudands	ultic fulvudands	D	H	E	G	DHEG
andisols	udands	fulvudands	eutric pachic fulvudands	D	H	E	H	DHEH
andisols	udands	fulvudands	eutric fulvudands	D	H	E	I	DHEI
andisols	udands	fulvudands	pachic fulvudands	D	H	E	J	DHEJ
andisols	udands	fulvudands	thaptic fulvudands	D	H	E	K	DHEK
andisols	udands	fulvudands	typic fulvudands	D	H	E	L	DHEL
andisols	udands	hapludands	lithic hapludands	D	H	F	A	DHFA
andisols	udands	hapludands	anthraquic hapludands	D	H	F	B	DHFB
andisols	udands	hapludands	aquic duric hapludands	D	H	F	C	DHFC
andisols	udands	hapludands	duric hapludands	D	H	F	D	DHFD
andisols	udands	hapludands	aquic hapludands	D	H	F	E	DHFE
andisols	udands	hapludands	oxyaquic hapludands	D	H	F	F	DHFF
andisols	udands	hapludands	alic hapludands	D	H	F	G	DHFG
andisols	udands	hapludands	acrudoxic hydric hapludands	D	H	F	H	DHFH
andisols	udands	hapludands	acrudoxic thaptic hapludands	D	H	F	I	DHFI
andisols	udands	hapludands	acrudoxic ultic hapludands	D	H	F	J	DHFJ
andisols	udands	hapludands	acrudoxic hapludands	D	H	F	K	DHFK
andisols	udands	hapludands	vitric hapludands	D	H	F	L	DHFL
andisols	udands	hapludands	hydric thaptic hapludands	D	H	F	M	DHFM
andisols	udands	hapludands	hydric hapludands	D	H	F	N	DHFN
andisols	udands	hapludands	eutric thaptic hapludands	D	H	F	O	DHFO
andisols	udands	hapludands	thaptic hapludands	D	H	F	P	DHFP
andisols	udands	hapludands	eutric hapludands	D	H	F	Q	DHFQ
andisols	udands	hapludands	oxic hapludands	D	H	F	R	DHFR
andisols	udands	hapludands	ultic hapludands	D	H	F	S	DHFS
andisols	udands	hapludands	alfic hapludands	D	H	F	T	DHFT
andisols	udands	hapludands	typic hapludands	D	H	F	U	DHFU
oxisols	aquox	acraquox	plinthic acraquox	E	A	A	A	EAAA
oxisols	aquox	acraquox	aeric acraquox	E	A	A	B	EAAB
oxisols	aquox	acraquox	typic acraquox	E	A	A	C	EAAC
oxisols	aquox	plinthaquox	aeric plinthaquox	E	A	B	A	EABA
oxisols	aquox	plinthaquox	typic plinthaquox	E	A	B	B	EABB
oxisols	aquox	eutraquox	histic eutraquox	E	A	C	A	EACA
oxisols	aquox	eutraquox	plinthic eutraquox	E	A	C	B	EACB
oxisols	aquox	eutraquox	aeric eutraquox	E	A	C	C	EACC

oxisols	aquox	eutraquox	humic eutraquox	E	A	C	D	EACD
oxisols	aquox	eutraquox	typic eutraquox	E	A	C	E	EACE
oxisols	aquox	haplaquox	histic haplaquox	E	A	D	A	EADA
oxisols	aquox	haplaquox	plinthic haplaquox	E	A	D	B	EADB
oxisols	aquox	haplaquox	aeric haplaquox	E	A	D	C	EADC
oxisols	aquox	haplaquox	humic haplaquox	E	A	D	D	EADD
oxisols	aquox	haplaquox	typic haplaquox	E	A	D	E	EADE
oxisols	torrox	acrotorrox	petroferric acrotorrox	E	B	A	A	EBAA
oxisols	torrox	acrotorrox	lithic acrotorrox	E	B	A	B	EBAB
oxisols	torrox	acrotorrox	typic acrotorrox	E	B	A	C	EBAC
oxisols	torrox	eutrotorrox	petroferric eutrotorrox	E	B	B	A	EBBA
oxisols	torrox	eutrotorrox	lithic eutrotorrox	E	B	B	B	EBBB
oxisols	torrox	eutrotorrox	typic eutrotorrox	E	B	B	C	EBBC
oxisols	torrox	haplotorrox	petroferric haplotorrox	E	B	C	A	EBCA
oxisols	torrox	haplotorrox	lithic haplotorrox	E	B	C	B	EBCB
oxisols	torrox	haplotorrox	typic haplotorrox	E	B	C	C	EBCC
oxisols	ustox	sombriustox	petroferric sombriustox	E	C	A	A	ECAA
oxisols	ustox	sombriustox	lithic sombriustox	E	C	A	B	ECAB
oxisols	ustox	sombriustox	humic sombriustox	E	C	A	C	ECAC
oxisols	ustox	sombriustox	typic sombriustox	E	C	A	D	ECAD
oxisols	ustox	acrustox	aquic petroferric acrustox	E	C	B	A	ECBA
oxisols	ustox	acrustox	petroferric acrustox	E	C	B	B	ECBB
oxisols	ustox	acrustox	aquic lithic acrustox	E	C	B	C	ECBC
oxisols	ustox	acrustox	lithic acrustox	E	C	B	D	ECBD
oxisols	ustox	acrustox	anionic aquic acrustox	E	C	B	E	ECBE
oxisols	ustox	acrustox	anionic acrustox	E	C	B	F	ECBF
oxisols	ustox	acrustox	plinthic acrustox	E	C	B	G	ECBG
oxisols	ustox	acrustox	aquic acrustox	E	C	B	H	ECBH
oxisols	ustox	acrustox	eutric acrustox	E	C	B	I	ECBI
oxisols	ustox	acrustox	humic rhodic acrustox	E	C	B	J	ECBJ
oxisols	ustox	acrustox	humic xanthic acrustox	E	C	B	K	ECBK
oxisols	ustox	acrustox	humic acrustox	E	C	B	L	ECBL
oxisols	ustox	acrustox	rhodic acrustox	E	C	B	M	ECBM
oxisols	ustox	acrustox	xanthic acrustox	E	C	B	N	ECBN
oxisols	ustox	acrustox	typic acrustox	E	C	B	O	ECBO
oxisols	ustox	eutrustox	aquic petroferric eutrustox	E	C	C	A	ECCA
oxisols	ustox	eutrustox	petroferric eutrustox	E	C	C	B	ECCB
oxisols	ustox	eutrustox	aquic lithic eutrustox	E	C	C	C	ECCC
oxisols	ustox	eutrustox	lithic eutrustox	E	C	C	D	ECCD
oxisols	ustox	eutrustox	plinthaquic eutrustox	E	C	C	E	ECCE
oxisols	ustox	eutrustox	plinthic eutrustox	E	C	C	F	ECCF
oxisols	ustox	eutrustox	aquic eutrustox	E	C	C	G	ECCG
oxisols	ustox	eutrustox	kandiustalfic eutrustox	E	C	C	H	ECCH
oxisols	ustox	eutrustox	humic inceptic eutrustox	E	C	C	I	ECCI
oxisols	ustox	eutrustox	inceptic eutrustox	E	C	C	J	ECCJ
oxisols	ustox	eutrustox	humic rhodic eutrustox	E	C	C	K	ECCK
oxisols	ustox	eutrustox	humic xanthic eutrustox	E	C	C	L	ECCL
oxisols	ustox	eutrustox	humic eutrustox	E	C	C	M	ECCM
oxisols	ustox	eutrustox	rhodic eutrustox	E	C	C	N	ECCN
oxisols	ustox	eutrustox	xanthic eutrustox	E	C	C	O	ECCO
oxisols	ustox	eutrustox	typic eutrustox	E	C	C	P	ECCP
oxisols	ustox	kandiustox	aquic petroferric kandiustox	E	C	D	A	ECDA
oxisols	ustox	kandiustox	petroferric kandiustox	E	C	D	B	ECDB
oxisols	ustox	kandiustox	aquic lithic kandiustox	E	C	D	C	ECDC
oxisols	ustox	kandiustox	lithic kandiustox	E	C	D	D	ECDD
oxisols	ustox	kandiustox	plinthaquic kandiustox	E	C	D	E	ECDE
oxisols	ustox	kandiustox	plinthic kandiustox	E	C	D	F	ECDF
oxisols	ustox	kandiustox	aquic kandiustox	E	C	D	G	ECDG
oxisols	ustox	kandiustox	humic rhodic kandiustox	E	C	D	H	ECDH
oxisols	ustox	kandiustox	humic xanthic kandiustox	E	C	D	I	ECDI

oxisols	ustox	kandiustox	humic kandiustox	E	C	D	J	ECDJ
oxisols	ustox	kandiustox	rhodic kandiustox	E	C	D	K	ECDK
oxisols	ustox	kandiustox	xanthic kandiustox	E	C	D	L	ECDL
oxisols	ustox	kandiustox	typic kandiustox	E	C	D	M	ECDM
oxisols	ustox	haplustox	aquic petroferric haplustox	E	C	E	A	ECEA
oxisols	ustox	haplustox	petroferric haplustox	E	C	E	B	ECEB
oxisols	ustox	haplustox	aquic lithic haplustox	E	C	E	C	ECEC
oxisols	ustox	haplustox	lithic haplustox	E	C	E	D	ECED
oxisols	ustox	haplustox	plinthaquic haplustox	E	C	E	E	ECEE
oxisols	ustox	haplustox	plinthic haplustox	E	C	E	F	ECEF
oxisols	ustox	haplustox	aqueptic haplustox	E	C	E	G	ECEG
oxisols	ustox	haplustox	aquic haplustox	E	C	E	H	ECEH
oxisols	ustox	haplustox	oxyaquic haplustox	E	C	E	I	ECEI
oxisols	ustox	haplustox	inceptic haplustox	E	C	E	J	ECEJ
oxisols	ustox	haplustox	humic rhodic haplustox	E	C	E	K	ECEK
oxisols	ustox	haplustox	humic xanthic haplustox	E	C	E	L	ECEL
oxisols	ustox	haplustox	humic haplustox	E	C	E	M	ECEM
oxisols	ustox	haplustox	rhodic haplustox	E	C	E	N	ECEN
oxisols	ustox	haplustox	xanthic haplustox	E	C	E	O	ECEO
oxisols	ustox	haplustox	typic haplustox	E	C	E	P	ECEP
oxisols	perox	sombriperox	petroferric sombriperox	E	D	A	A	EDAA
oxisols	perox	sombriperox	lithic sombriperox	E	D	A	B	EDAB
oxisols	perox	sombriperox	humic sombriperox	E	D	A	C	EDAC
oxisols	perox	sombriperox	typic sombriperox	E	D	A	D	EDAD
oxisols	perox	acroperox	aquic petroferric acroperox	E	D	B	A	EDBA
oxisols	perox	acroperox	petroferric acroperox	E	D	B	B	EDBB
oxisols	perox	acroperox	aquic lithic acroperox	E	D	B	C	EDBC
oxisols	perox	acroperox	lithic acroperox	E	D	B	D	EDBD
oxisols	perox	acroperox	anionic acroperox	E	D	B	E	EDBE
oxisols	perox	acroperox	plinthic acroperox	E	D	B	F	EDBF
oxisols	perox	acroperox	aquic acroperox	E	D	B	G	EDBG
oxisols	perox	acroperox	humic rhodic acroperox	E	D	B	H	EDBH
oxisols	perox	acroperox	humic xanthic acroperox	E	D	B	I	EDBI
oxisols	perox	acroperox	humic acroperox	E	D	B	J	EDBJ
oxisols	perox	acroperox	rhodic acroperox	E	D	B	K	EDBK
oxisols	perox	acroperox	xanthic acroperox	E	D	B	L	EDBL
oxisols	perox	acroperox	typic acroperox	E	D	B	M	EDBM
oxisols	perox	eutroperox	aquic petroferric eutroperox	E	D	C	A	EDCA
oxisols	perox	eutroperox	petroferric eutroperox	E	D	C	B	EDCB
oxisols	perox	eutroperox	aquic lithic eutroperox	E	D	C	C	EDCC
oxisols	perox	eutroperox	lithic eutroperox	E	D	C	D	EDCD
oxisols	perox	eutroperox	plinthaquic eutroperox	E	D	C	E	EDCE
oxisols	perox	eutroperox	plinthic eutroperox	E	D	C	F	EDCF
oxisols	perox	eutroperox	aquic eutroperox	E	D	C	G	EDCG
oxisols	perox	eutroperox	kandiudalfic eutroperox	E	D	C	H	EDCH
oxisols	perox	eutroperox	humic inceptic eutroperox	E	D	C	I	EDCI
oxisols	perox	eutroperox	inceptic eutroperox	E	D	C	J	EDCJ
oxisols	perox	eutroperox	humic rhodic eutroperox	E	D	C	K	EDCK
oxisols	perox	eutroperox	humic xanthic eutroperox	E	D	C	L	EDCL
oxisols	perox	eutroperox	humic eutroperox	E	D	C	M	EDCM
oxisols	perox	eutroperox	rhodic eutroperox	E	D	C	N	EDCN
oxisols	perox	eutroperox	xanthic eutroperox	E	D	C	O	EDCO
oxisols	perox	eutroperox	typic eutroperox	E	D	C	P	EDCP
oxisols	perox	kandiperox	aquic petroferric kandiperox	E	D	D	A	EDDA
oxisols	perox	kandiperox	petroferric kandiperox	E	D	D	B	EDDB
oxisols	perox	kandiperox	aquic lithic kandiperox	E	D	D	C	EDDC
oxisols	perox	kandiperox	lithic kandiperox	E	D	D	D	EDDD
oxisols	perox	kandiperox	plinthaquic kandiperox	E	D	D	E	EDDE
oxisols	perox	kandiperox	plinthic kandiperox	E	D	D	F	EDDF
oxisols	perox	kandiperox	aquic kandiperox	E	D	D	G	EDDG

oxisols	perox	kandiperox	andic kandiperox	E	D	D	H	EDDH
oxisols	perox	kandiperox	humic rhodic kandiperox	E	D	D	I	EDDI
oxisols	perox	kandiperox	humic xanthic kandiperox	E	D	D	J	EDDJ
oxisols	perox	kandiperox	humic kandiperox	E	D	D	K	EDDK
oxisols	perox	kandiperox	rhodic kandiperox	E	D	D	L	EDDL
oxisols	perox	kandiperox	xanthic kandiperox	E	D	D	M	EDDM
oxisols	perox	kandiperox	typic kandiperox	E	D	D	N	EDDN
oxisols	perox	haploperox	aquic petroferric haploperox	E	D	E	A	EDEA
oxisols	perox	haploperox	petroferric haploperox	E	D	E	B	EDEB
oxisols	perox	haploperox	aquic lithic haploperox	E	D	E	C	EDEC
oxisols	perox	haploperox	lithic haploperox	E	D	E	D	EDED
oxisols	perox	haploperox	plinthaquic haploperox	E	D	E	E	EDEE
oxisols	perox	haploperox	plinthic haploperox	E	D	E	F	EDEF
oxisols	perox	haploperox	aquic haploperox	E	D	E	G	EDEG
oxisols	perox	haploperox	andic haploperox	E	D	E	H	EDEH
oxisols	perox	haploperox	humic rhodic haploperox	E	D	E	I	EDEI
oxisols	perox	haploperox	humic xanthic haploperox	E	D	E	J	EDEJ
oxisols	perox	haploperox	humic haploperox	E	D	E	K	EDEK
oxisols	perox	haploperox	rhodic haploperox	E	D	E	L	EDEL
oxisols	perox	haploperox	xanthic haploperox	E	D	E	M	EDEM
oxisols	perox	haploperox	typic haploperox	E	D	E	N	EDEN
oxisols	udox	sombriudox	petroferric sombriudox	E	E	A	A	EEAA
oxisols	udox	sombriudox	lithic sombriudox	E	E	A	B	EEAB
oxisols	udox	sombriudox	humic sombriudox	E	E	A	C	EEAC
oxisols	udox	sombriudox	typic sombriudox	E	E	A	D	EEAD
oxisols	udox	acrudox	aquic petroferric acrudox	E	E	B	A	EEBA
oxisols	udox	acrudox	petroferric acrudox	E	E	B	B	EEBB
oxisols	udox	acrudox	aquic lithic acrudox	E	E	B	C	EEBC
oxisols	udox	acrudox	lithic acrudox	E	E	B	D	EEBD
oxisols	udox	acrudox	anionic aquic acrudox	E	E	B	E	EEBE
oxisols	udox	acrudox	anionic acrudox	E	E	B	F	EEBF
oxisols	udox	acrudox	plinthic acrudox	E	E	B	G	EEBG
oxisols	udox	acrudox	aquic acrudox	E	E	B	H	EEBH
oxisols	udox	acrudox	eutric acrudox	E	E	B	I	EEBI
oxisols	udox	acrudox	humic rhodic acrudox	E	E	B	J	EEBJ
oxisols	udox	acrudox	humic xanthic acrudox	E	E	B	K	EEBK
oxisols	udox	acrudox	humic acrudox	E	E	B	L	EEBL
oxisols	udox	acrudox	rhodic acrudox	E	E	B	M	EEBM
oxisols	udox	acrudox	xanthic acrudox	E	E	B	N	EEBN
oxisols	udox	acrudox	typic acrudox	E	E	B	O	EEBO
oxisols	udox	eutrudox	aquic petroferric eutrudox	E	E	C	A	EECA
oxisols	udox	eutrudox	petroferric eutrudox	E	E	C	B	EECB
oxisols	udox	eutrudox	aquic lithic eutrudox	E	E	C	C	EECC
oxisols	udox	eutrudox	lithic eutrudox	E	E	C	D	EECD
oxisols	udox	eutrudox	plinthaquic eutrudox	E	E	C	E	EECE
oxisols	udox	eutrudox	plinthic eutrudox	E	E	C	F	EECF
oxisols	udox	eutrudox	aquic eutrudox	E	E	C	G	EECG
oxisols	udox	eutrudox	kandiudalfic eutrudox	E	E	C	H	EECH
oxisols	udox	eutrudox	humic inceptic eutrudox	E	E	C	I	EECI
oxisols	udox	eutrudox	inceptic eutrudox	E	E	C	J	EECJ
oxisols	udox	eutrudox	humic rhodic eutrudox	E	E	C	K	EECK
oxisols	udox	eutrudox	humic xanthic eutrudox	E	E	C	L	EECL
oxisols	udox	eutrudox	humic eutrudox	E	E	C	M	EECM
oxisols	udox	eutrudox	rhodic eutrudox	E	E	C	N	EECN
oxisols	udox	eutrudox	xanthic eutrudox	E	E	C	O	EECO
oxisols	udox	eutrudox	typic eutrudox	E	E	C	P	EECP
oxisols	udox	kandiudox	aquic petroferric kandiudox	E	E	D	A	EEDA
oxisols	udox	kandiudox	petroferric kandiudox	E	E	D	B	EEDB
oxisols	udox	kandiudox	aquic lithic kandiudox	E	E	D	C	EEDC
oxisols	udox	kandiudox	lithic kandiudox	E	E	D	D	EEDD

oxisols	udox	kandiudox	plinthaquic kandiudox	E	E	D	E	EEDE
oxisols	udox	kandiudox	plinthic kandiudox	E	E	D	F	EEDF
oxisols	udox	kandiudox	aquic kandiudox	E	E	D	G	EEDG
oxisols	udox	kandiudox	andic kandiudox	E	E	D	H	EEDH
oxisols	udox	kandiudox	humic rhodic kandiudox	E	E	D	I	EEDI
oxisols	udox	kandiudox	humic xanthic kandiudox	E	E	D	J	EEDJ
oxisols	udox	kandiudox	humic kandiudox	E	E	D	K	EEDK
oxisols	udox	kandiudox	rhodic kandiudox	E	E	D	L	EEDL
oxisols	udox	kandiudox	xanthic kandiudox	E	E	D	M	EEDM
oxisols	udox	kandiudox	typic kandiudox	E	E	D	N	EEDN
oxisols	udox	hapludox	aquic petroferric hapludox	E	E	E	A	EEEA
oxisols	udox	hapludox	petroferric hapludox	E	E	E	B	EEEB
oxisols	udox	hapludox	aquic lithic hapludox	E	E	E	C	EEEC
oxisols	udox	hapludox	lithic hapludox	E	E	E	D	EEED
oxisols	udox	hapludox	plinthaquic hapludox	E	E	E	E	EEEE
oxisols	udox	hapludox	plinthic hapludox	E	E	E	F	EEEF
oxisols	udox	hapludox	aquic hapludox	E	E	E	G	EEEG
oxisols	udox	hapludox	inceptic hapludox	E	E	E	H	EEEH
oxisols	udox	hapludox	andic hapludox	E	E	E	I	EEEI
oxisols	udox	hapludox	humic rhodic hapludox	E	E	E	J	EEEJ
oxisols	udox	hapludox	humic xanthic hapludox	E	E	E	K	EEEK
oxisols	udox	hapludox	humic hapludox	E	E	E	L	EEEL
oxisols	udox	hapludox	rhodic hapludox	E	E	E	M	EEEM
oxisols	udox	hapludox	xanthic hapludox	E	E	E	N	EEEN
oxisols	udox	hapludox	typic hapludox	E	E	E	O	EEEO
vertisols	aquerts	sulfaquerts	salic sulfaquerts	F	A	A	A	FAAA
vertisols	aquerts	sulfaquerts	sulfic sulfaquerts	F	A	A	B	FAAB
vertisols	aquerts	sulfaquerts	typic sulfaquerts	F	A	A	C	FAAC
vertisols	aquerts	salaquerts	aridic salaquerts	F	A	B	A	FABA
vertisols	aquerts	salaquerts	ustic salaquerts	F	A	B	B	FABB
vertisols	aquerts	salaquerts	leptic salaquerts	F	A	B	C	FABC
vertisols	aquerts	salaquerts	entic salaquerts	F	A	B	D	FABD
vertisols	aquerts	salaquerts	chromic salaquerts	F	A	B	E	FABE
vertisols	aquerts	salaquerts	typic salaquerts	F	A	B	F	FABF
vertisols	aquerts	duraquerts	aridic duraquerts	F	A	C	A	FACA
vertisols	aquerts	duraquerts	xeric duraquerts	F	A	C	B	FACB
vertisols	aquerts	duraquerts	ustic duraquerts	F	A	C	C	FACC
vertisols	aquerts	duraquerts	aeric duraquerts	F	A	C	D	FACD
vertisols	aquerts	duraquerts	chromic duraquerts	F	A	C	E	FACE
vertisols	aquerts	duraquerts	typic duraquerts	F	A	C	F	FACF
vertisols	aquerts	natraquerts	typic natraquerts	F	A	D	A	FADA
vertisols	aquerts	calciaquerts	aeric calciaquerts	F	A	E	A	FAEA
vertisols	aquerts	calciaquerts	typic calciaquerts	F	A	E	B	FAEB
vertisols	aquerts	dystraquerts	sulfaqueptic dystraquerts	F	A	F	A	FAFA
vertisols	aquerts	dystraquerts	aridic dystraquerts	F	A	F	B	FAFB
vertisols	aquerts	dystraquerts	ustic dystraquerts	F	A	F	C	FAFC
vertisols	aquerts	dystraquerts	aeric dystraquerts	F	A	F	D	FAFD
vertisols	aquerts	dystraquerts	leptic dystraquerts	F	A	F	E	FAFE
vertisols	aquerts	dystraquerts	entic dystraquerts	F	A	F	F	FAFF
vertisols	aquerts	dystraquerts	chromic dystraquerts	F	A	F	G	FAFG
vertisols	aquerts	dystraquerts	typic dystraquerts	F	A	F	H	FAFH
vertisols	aquerts	epiaquerts	halic epiaquerts	F	A	G	A	FAGA
vertisols	aquerts	epiaquerts	sodic epiaquerts	F	A	G	B	FAGB
vertisols	aquerts	epiaquerts	aridic epiaquerts	F	A	G	C	FAGC
vertisols	aquerts	epiaquerts	xeric epiaquerts	F	A	G	D	FAGD
vertisols	aquerts	epiaquerts	ustic epiaquerts	F	A	G	E	FAGE
vertisols	aquerts	epiaquerts	aeric epiaquerts	F	A	G	F	FAGF
vertisols	aquerts	epiaquerts	leptic epiaquerts	F	A	G	G	FAGG
vertisols	aquerts	epiaquerts	entic epiaquerts	F	A	G	H	FAGH
vertisols	aquerts	epiaquerts	chromic epiaquerts	F	A	G	I	FAGI

vertisols	aquerts	epiaquerts	typic epiaquerts	F	A	G	J	FAGJ
vertisols	aquerts	endoaquerts	halic endoaquerts	F	A	H	A	FAHA
vertisols	aquerts	endoaquerts	sodic endoaquerts	F	A	H	B	FAHB
vertisols	aquerts	endoaquerts	aridic endoaquerts	F	A	H	C	FAHC
vertisols	aquerts	endoaquerts	xeric endoaquerts	F	A	H	D	FAHD
vertisols	aquerts	endoaquerts	ustic endoaquerts	F	A	H	E	FAHE
vertisols	aquerts	endoaquerts	aeric endoaquerts	F	A	H	F	FAHF
vertisols	aquerts	endoaquerts	leptic endoaquerts	F	A	H	G	FAHG
vertisols	aquerts	endoaquerts	entic endoaquerts	F	A	H	H	FAHH
vertisols	aquerts	endoaquerts	chromic endoaquerts	F	A	H	I	FAHI
vertisols	aquerts	endoaquerts	typic endoaquerts	F	A	H	J	FAHJ
vertisols	cryerts	humicryerts	sodic humicryerts	F	B	A	A	FBAA
vertisols	cryerts	humicryerts	typic humicryerts	F	B	A	B	FBAB
vertisols	cryerts	haplocryerts	sodic haplocryerts	F	B	B	A	FBBA
vertisols	cryerts	haplocryerts	chromic haplocryerts	F	B	B	B	FBBB
vertisols	cryerts	haplocryerts	typic haplocryerts	F	B	B	C	FBBC
vertisols	xererts	durixererts	halic durixererts	F	C	A	A	FCAA
vertisols	xererts	durixererts	sodic durixererts	F	C	A	B	FCAB
vertisols	xererts	durixererts	aquic durixererts	F	C	A	C	FCAC
vertisols	xererts	durixererts	aridic durixererts	F	C	A	D	FCAD
vertisols	xererts	durixererts	udic durixererts	F	C	A	E	FCAE
vertisols	xererts	durixererts	haplic durixererts	F	C	A	F	FCAF
vertisols	xererts	durixererts	chromic durixererts	F	C	A	G	FCAG
vertisols	xererts	durixererts	typic durixererts	F	C	A	H	FCAH
vertisols	xererts	calcixererts	lithic calcixererts	F	C	B	A	FCBA
vertisols	xererts	calcixererts	petrocalcic calcixererts	F	C	B	B	FCBB
vertisols	xererts	calcixererts	aridic calcixererts	F	C	B	C	FCBC
vertisols	xererts	calcixererts	leptic calcixererts	F	C	B	D	FCBD
vertisols	xererts	calcixererts	entic calcixererts	F	C	B	E	FCBE
vertisols	xererts	calcixererts	chromic calcixererts	F	C	B	F	FCBF
vertisols	xererts	calcixererts	typic calcixererts	F	C	B	G	FCBG
vertisols	xererts	haploxererts	lithic haploxererts	F	C	C	A	FCCA
vertisols	xererts	haploxererts	halic haploxererts	F	C	C	B	FCCB
vertisols	xererts	haploxererts	sodic haploxererts	F	C	C	C	FCCC
vertisols	xererts	haploxererts	aridic haploxererts	F	C	C	D	FCCD
vertisols	xererts	haploxererts	aquic haploxererts	F	C	C	E	FCCE
vertisols	xererts	haploxererts	udic haploxererts	F	C	C	F	FCCF
vertisols	xererts	haploxererts	leptic haploxererts	F	C	C	G	FCCG
vertisols	xererts	haploxererts	entic haploxererts	F	C	C	H	FCCH
vertisols	xererts	haploxererts	chromic haploxererts	F	C	C	I	FCCI
vertisols	xererts	haploxererts	typic haploxererts	F	C	C	J	FCCJ
vertisols	torrerts	salitorrerts	aquic salitorrerts	F	D	A	A	FDAA
vertisols	torrerts	salitorrerts	leptic salitorrerts	F	D	A	B	FDAB
vertisols	torrerts	salitorrerts	entic salitorrerts	F	D	A	C	FDAC
vertisols	torrerts	salitorrerts	chromic salitorrerts	F	D	A	D	FDAD
vertisols	torrerts	salitorrerts	typic salitorrerts	F	D	A	E	FDAE
vertisols	torrerts	gypsitorrerts	chromic gypsitorrerts	F	D	B	A	FDBA
vertisols	torrerts	gypsitorrerts	typic gypsitorrerts	F	D	B	B	FDBB
vertisols	torrerts	calcitorrerts	petrocalcic calcitorrerts	F	D	C	A	FDCA
vertisols	torrerts	calcitorrerts	leptic calcitorrerts	F	D	C	B	FDCB
vertisols	torrerts	calcitorrerts	entic calcitorrerts	F	D	C	C	FDCC
vertisols	torrerts	calcitorrerts	chromic calcitorrerts	F	D	C	D	FDCD
vertisols	torrerts	calcitorrerts	typic calcitorrerts	F	D	C	E	FDCE
vertisols	torrerts	haplotorrerts	halic haplotorrerts	F	D	D	A	FDDA
vertisols	torrerts	haplotorrerts	sodic haplotorrerts	F	D	D	B	FDDB
vertisols	torrerts	haplotorrerts	leptic haplotorrerts	F	D	D	C	FDDC
vertisols	torrerts	haplotorrerts	entic haplotorrerts	F	D	D	D	FDDD
vertisols	torrerts	haplotorrerts	chromic haplotorrerts	F	D	D	E	FDDE
vertisols	torrerts	haplotorrerts	typic haplotorrerts	F	D	D	F	FDDF
vertisols	usterts	dystrusterts	lithic dystrusterts	F	E	A	A	FEAA

vertisols	usterts	dystrusterts	aquic dystrusterts	F	E	A	B	FEAB
vertisols	usterts	dystrusterts	aridic dystrusterts	F	E	A	C	FEAC
vertisols	usterts	dystrusterts	udic dystrusterts	F	E	A	D	FEAD
vertisols	usterts	dystrusterts	leptic dystrusterts	F	E	A	E	FEAE
vertisols	usterts	dystrusterts	entic dystrusterts	F	E	A	F	FEAF
vertisols	usterts	dystrusterts	chromic dystrusterts	F	E	A	G	FEAG
vertisols	usterts	dystrusterts	typic dystrusterts	F	E	A	H	FEAH
vertisols	usterts	salusterts	lithic salusterts	F	E	B	A	FEBA
vertisols	usterts	salusterts	sodic salusterts	F	E	B	B	FEBB
vertisols	usterts	salusterts	aquic salusterts	F	E	B	C	FEBC
vertisols	usterts	salusterts	aridic salusterts	F	E	B	D	FEBD
vertisols	usterts	salusterts	leptic salusterts	F	E	B	E	FEBE
vertisols	usterts	salusterts	entic salusterts	F	E	B	F	FEBF
vertisols	usterts	salusterts	chromic salusterts	F	E	B	G	FEBG
vertisols	usterts	salusterts	typic salusterts	F	E	B	H	FEBH
vertisols	usterts	gypsiusterts	lithic gypsiusterts	F	E	C	A	FECA
vertisols	usterts	gypsiusterts	halic gypsiusterts	F	E	C	B	FECB
vertisols	usterts	gypsiusterts	sodic gypsiusterts	F	E	C	C	FECC
vertisols	usterts	gypsiusterts	aridic gypsiusterts	F	E	C	D	FECD
vertisols	usterts	gypsiusterts	udic gypsiusterts	F	E	C	E	FECE
vertisols	usterts	gypsiusterts	leptic gypsiusterts	F	E	C	F	FECF
vertisols	usterts	gypsiusterts	entic gypsiusterts	F	E	C	G	FECG
vertisols	usterts	gypsiusterts	chromic gypsiusterts	F	E	C	H	FECH
vertisols	usterts	gypsiusterts	typic gypsiusterts	F	E	C	I	FECI
vertisols	usterts	calciusterts	lithic calciusterts	F	E	D	A	FEDA
vertisols	usterts	calciusterts	halic calciusterts	F	E	D	B	FEDB
vertisols	usterts	calciusterts	sodic calciusterts	F	E	D	C	FEDC
vertisols	usterts	calciusterts	petrocalcic calciusterts	F	E	D	D	FEDD
vertisols	usterts	calciusterts	aridic calciusterts	F	E	D	E	FEDE
vertisols	usterts	calciusterts	udic calciusterts	F	E	D	F	FEDF
vertisols	usterts	calciusterts	leptic calciusterts	F	E	D	G	FEDG
vertisols	usterts	calciusterts	entic calciusterts	F	E	D	H	FEDH
vertisols	usterts	calciusterts	chromic calciusterts	F	E	D	I	FEDI
vertisols	usterts	calciusterts	typic calciusterts	F	E	D	J	FEDJ
vertisols	usterts	haplusterts	lithic haplusterts	F	E	E	A	FEEA
vertisols	usterts	haplusterts	halic haplusterts	F	E	E	B	FEEB
vertisols	usterts	haplusterts	sodic haplusterts	F	E	E	C	FEEC
vertisols	usterts	haplusterts	petrocalcic haplusterts	F	E	E	D	FEED
vertisols	usterts	haplusterts	gypsic haplusterts	F	E	E	E	FEEE
vertisols	usterts	haplusterts	calcic haplusterts	F	E	E	F	FEEF
vertisols	usterts	haplusterts	aridic leptic haplusterts	F	E	E	G	FEEG
vertisols	usterts	haplusterts	aridic haplusterts	F	E	E	H	FEEH
vertisols	usterts	haplusterts	leptic udic haplusterts	F	E	E	I	FEEI
vertisols	usterts	haplusterts	entic udic haplusterts	F	E	E	J	FEEJ
vertisols	usterts	haplusterts	chromic udic haplusterts	F	E	E	K	FEEK
vertisols	usterts	haplusterts	udic haplusterts	F	E	E	L	FEEL
vertisols	usterts	haplusterts	leptic haplusterts	F	E	E	M	FEEM
vertisols	usterts	haplusterts	entic haplusterts	F	E	E	N	FEEN
vertisols	usterts	haplusterts	chromic haplusterts	F	E	E	O	FEEO
vertisols	usterts	haplusterts	typic haplusterts	F	E	E	P	FEEP
vertisols	uderts	dystruderts	aquic dystruderts	F	F	A	A	FFAA
vertisols	uderts	dystruderts	oxyaquic dystruderts	F	F	A	B	FFAB
vertisols	uderts	dystruderts	leptic dystruderts	F	F	A	C	FFAC
vertisols	uderts	dystruderts	entic dystruderts	F	F	A	D	FFAD
vertisols	uderts	dystruderts	chromic dystruderts	F	F	A	E	FFAE
vertisols	uderts	dystruderts	typic dystruderts	F	F	A	F	FFAF
vertisols	uderts	hapluderts	lithic hapluderts	F	F	B	A	FFBA
vertisols	uderts	hapluderts	aquic hapluderts	F	F	B	B	FFBB
vertisols	uderts	hapluderts	oxyaquic hapluderts	F	F	B	C	FFBC
vertisols	uderts	hapluderts	leptic hapluderts	F	F	B	D	FFBD

vertisols	uderts	hapluderts	entic hapluderts	F	F	B	E	FFBE
vertisols	uderts	hapluderts	chromic hapluderts	F	F	B	F	FFBF
vertisols	uderts	hapluderts	typic hapluderts	F	F	B	G	FFBG
aridisols	cryids	salicryids	aquic salicryids	G	A	A	A	GAAA
aridisols	cryids	salicryids	typic salicryids	G	A	A	B	GAAB
aridisols	cryids	petrocryids	xereptic petrocryids	G	A	B	A	GABA
aridisols	cryids	petrocryids	duric xeric petrocryids	G	A	B	B	GABB
aridisols	cryids	petrocryids	duric petrocryids	G	A	B	C	GABC
aridisols	cryids	petrocryids	petrogypsic petrocryids	G	A	B	D	GABD
aridisols	cryids	petrocryids	xeric petrocryids	G	A	B	E	GABE
aridisols	cryids	petrocryids	ustic petrocryids	G	A	B	F	GABF
aridisols	cryids	petrocryids	typic petrocryids	G	A	B	G	GABG
aridisols	cryids	gypsicryids	calcic gypsicryids	G	A	C	A	GACA
aridisols	cryids	gypsicryids	vitrixerandic gypsicryids	G	A	C	B	GACB
aridisols	cryids	gypsicryids	vitrandic gypsicryids	G	A	C	C	GACC
aridisols	cryids	gypsicryids	typic gypsicryids	G	A	C	D	GACD
aridisols	cryids	argicryids	lithic argicryids	G	A	D	A	GADA
aridisols	cryids	argicryids	vertic argicryids	G	A	D	B	GADB
aridisols	cryids	argicryids	natric argicryids	G	A	D	C	GADC
aridisols	cryids	argicryids	vitrixerandic argicryids	G	A	D	D	GADD
aridisols	cryids	argicryids	vitrandic argicryids	G	A	D	E	GADE
aridisols	cryids	argicryids	xeric argicryids	G	A	D	F	GADF
aridisols	cryids	argicryids	ustic argicryids	G	A	D	G	GADG
aridisols	cryids	argicryids	typic argicryids	G	A	D	H	GADH
aridisols	cryids	calcicryids	lithic calcicryids	G	A	E	A	GAEA
aridisols	cryids	calcicryids	vitrixerandic calcicryids	G	A	E	B	GAEB
aridisols	cryids	calcicryids	vitrandic calcicryids	G	A	E	C	GAEC
aridisols	cryids	calcicryids	xeric calcicryids	G	A	E	D	GAED
aridisols	cryids	calcicryids	ustic calcicryids	G	A	E	E	GAEE
aridisols	cryids	calcicryids	typic calcicryids	G	A	E	F	GAEF
aridisols	cryids	haplocryids	lithic haplocryids	G	A	F	A	GAFA
aridisols	cryids	haplocryids	vertic haplocryids	G	A	F	B	GAFB
aridisols	cryids	haplocryids	vitrixerandic haplocryids	G	A	F	C	GAFC
aridisols	cryids	haplocryids	vitrandic haplocryids	G	A	F	D	GAFD
aridisols	cryids	haplocryids	xeric haplocryids	G	A	F	E	GAFE
aridisols	cryids	haplocryids	ustic haplocryids	G	A	F	F	GAFF
aridisols	cryids	haplocryids	typic haplocryids	G	A	F	G	GAFG
aridisols	salids	aquisalids	anhydritic aquisalids	G	B	A	A	GBAA
aridisols	salids	aquisalids	gypsic aquisalids	G	B	A	B	GBAB
aridisols	salids	aquisalids	calcic aquisalids	G	B	A	C	GBAC
aridisols	salids	aquisalids	typic aquisalids	G	B	A	D	GBAD
aridisols	salids	haplosalids	duric haplosalids	G	B	B	A	GBBA
aridisols	salids	haplosalids	petrogypsic haplosalids	G	B	B	B	GBBB
aridisols	salids	haplosalids	anhydritic haplosalids	G	B	B	C	GBBC
aridisols	salids	haplosalids	gypsic haplosalids	G	B	B	D	GBBD
aridisols	salids	haplosalids	calcic haplosalids	G	B	B	E	GBBE
aridisols	salids	haplosalids	typic haplosalids	G	B	B	F	GBBF
aridisols	durids	natridurids	vertic natridurids	G	C	A	A	GCAA
aridisols	durids	natridurids	aquic natrargidic natridurids	G	C	A	B	GCAB
aridisols	durids	natridurids	aquic natridurids	G	C	A	C	GCAC
aridisols	durids	natridurids	natrixeralfic natridurids	G	C	A	D	GCAD
aridisols	durids	natridurids	natrargidic natridurids	G	C	A	E	GCAE
aridisols	durids	natridurids	vitrixerandic natridurids	G	C	A	F	GCAF
aridisols	durids	natridurids	vitrandic natridurids	G	C	A	G	GCAG
aridisols	durids	natridurids	xeric natridurids	G	C	A	H	GCAH
aridisols	durids	natridurids	typic natridurids	G	C	A	I	GCAI
aridisols	durids	argidurids	vertic argidurids	G	C	B	A	GCBA
aridisols	durids	argidurids	aquic argidurids	G	C	B	B	GCBB
aridisols	durids	argidurids	abruptic xeric argidurids	G	C	B	C	GCBC
aridisols	durids	argidurids	abruptic argidurids	G	C	B	D	GCBD

aridisols	durids	argidurids	haploxeralfic argidurids	G	C	B	E	GCBE
aridisols	durids	argidurids	argidic argidurids	G	C	B	F	GCBF
aridisols	durids	argidurids	vitrixerandic argidurids	G	C	B	G	GCBG
aridisols	durids	argidurids	vitrandic argidurids	G	C	B	H	GCBH
aridisols	durids	argidurids	xeric argidurids	G	C	B	I	GCBI
aridisols	durids	argidurids	ustic argidurids	G	C	B	J	GCBJ
aridisols	durids	argidurids	typic argidurids	G	C	B	K	GCBK
aridisols	durids	haplodurids	aquicambidic haplodurids	G	C	C	A	GCCA
aridisols	durids	haplodurids	aquic haplodurids	G	C	C	B	GCCB
aridisols	durids	haplodurids	xereptic haplodurids	G	C	C	C	GCCC
aridisols	durids	haplodurids	cambidic haplodurids	G	C	C	D	GCCD
aridisols	durids	haplodurids	vitrixerandic haplodurids	G	C	C	E	GCCE
aridisols	durids	haplodurids	vitrandic haplodurids	G	C	C	F	GCCF
aridisols	durids	haplodurids	xeric haplodurids	G	C	C	G	GCCG
aridisols	durids	haplodurids	ustic haplodurids	G	C	C	H	GCCH
aridisols	durids	haplodurids	typic haplodurids	G	C	C	I	GCCI
aridisols	gypsids	petrogypsids	petrocalcic petrogypsids	G	D	A	A	GDAA
aridisols	gypsids	petrogypsids	calcic petrogypsids	G	D	A	B	GDAB
aridisols	gypsids	petrogypsids	vitrixerandic petrogypsids	G	D	A	C	GDAC
aridisols	gypsids	petrogypsids	vitrandic petrogypsids	G	D	A	D	GDAD
aridisols	gypsids	petrogypsids	xeric petrogypsids	G	D	A	E	GDAE
aridisols	gypsids	petrogypsids	ustic petrogypsids	G	D	A	F	GDAF
aridisols	gypsids	petrogypsids	typic petrogypsids	G	D	A	G	GDAG
aridisols	gypsids	natrigypsids	lithic natrigypsids	G	D	B	A	GDBA
aridisols	gypsids	natrigypsids	vertic natrigypsids	G	D	B	B	GDBB
aridisols	gypsids	natrigypsids	petronodic natrigypsids	G	D	B	C	GDBC
aridisols	gypsids	natrigypsids	vitrixerandic natrigypsids	G	D	B	D	GDBD
aridisols	gypsids	natrigypsids	vitrandic natrigypsids	G	D	B	E	GDBE
aridisols	gypsids	natrigypsids	xeric natrigypsids	G	D	B	F	GDBF
aridisols	gypsids	natrigypsids	ustic natrigypsids	G	D	B	G	GDBG
aridisols	gypsids	natrigypsids	typic natrigypsids	G	D	B	H	GDBH
aridisols	gypsids	argigypsids	lithic argigypsids	G	D	C	A	GDCA
aridisols	gypsids	argigypsids	vertic argigypsids	G	D	C	B	GDCB
aridisols	gypsids	argigypsids	calcic argigypsids	G	D	C	C	GDCC
aridisols	gypsids	argigypsids	petronodic argigypsids	G	D	C	D	GDCD
aridisols	gypsids	argigypsids	vitrixerandic argigypsids	G	D	C	E	GDCE
aridisols	gypsids	argigypsids	vitrandic argigypsids	G	D	C	F	GDCF
aridisols	gypsids	argigypsids	xeric argigypsids	G	D	C	G	GDCG
aridisols	gypsids	argigypsids	ustic argigypsids	G	D	C	H	GDCH
aridisols	gypsids	argigypsids	typic argigypsids	G	D	C	I	GDCI
aridisols	gypsids	calcigypsids	lithic calcigypsids	G	D	D	A	GDDA
aridisols	gypsids	calcigypsids	petronodic calcigypsids	G	D	D	B	GDDB
aridisols	gypsids	calcigypsids	vitrixerandic calcigypsids	G	D	D	C	GDDC
aridisols	gypsids	calcigypsids	vitrandic calcigypsids	G	D	D	D	GDDD
aridisols	gypsids	calcigypsids	xeric calcigypsids	G	D	D	E	GDDE
aridisols	gypsids	calcigypsids	ustic calcigypsids	G	D	D	F	GDDF
aridisols	gypsids	calcigypsids	typic calcigypsids	G	D	D	G	GDDG
aridisols	gypsids	haplogypsids	lithic haplogypsids	G	D	E	A	GDEA
aridisols	gypsids	haplogypsids	leptic haplogypsids	G	D	E	B	GDEB
aridisols	gypsids	haplogypsids	sodic haplogypsids	G	D	E	C	GDEC
aridisols	gypsids	haplogypsids	petronodic haplogypsids	G	D	E	D	GDED
aridisols	gypsids	haplogypsids	vitrixerandic haplogypsids	G	D	E	E	GDEE
aridisols	gypsids	haplogypsids	vitrandic haplogypsids	G	D	E	F	GDEF
aridisols	gypsids	haplogypsids	xeric haplogypsids	G	D	E	G	GDEG
aridisols	gypsids	haplogypsids	ustic haplogypsids	G	D	E	H	GDEH
aridisols	gypsids	haplogypsids	typic haplogypsids	G	D	E	I	GDEI
aridisols	argids	petroargids	petrogypsic ustic petroargids	G	E	A	A	GEAA
aridisols	argids	petroargids	petrogypsic petroargids	G	E	A	B	GEAB
aridisols	argids	petroargids	duric xeric petroargids	G	E	A	C	GEAC
aridisols	argids	petroargids	duric petroargids	G	E	A	D	GEAD

aridisols	argids	petroargids	natric petroargids	G	E	A	E	GEAE
aridisols	argids	petroargids	xeric petroargids	G	E	A	F	GEAF
aridisols	argids	petroargids	ustic petroargids	G	E	A	G	GEAG
aridisols	argids	petroargids	typic petroargids	G	E	A	H	GEAH
aridisols	argids	natrargids	lithic xeric natrargids	G	E	B	A	GEBA
aridisols	argids	natrargids	lithic ustic natrargids	G	E	B	B	GEBB
aridisols	argids	natrargids	lithic natrargids	G	E	B	C	GEBC
aridisols	argids	natrargids	xerertic natrargids	G	E	B	D	GEBD
aridisols	argids	natrargids	ustertic natrargids	G	E	B	E	GEBE
aridisols	argids	natrargids	vertic natrargids	G	E	B	F	GEBF
aridisols	argids	natrargids	aquic natrargids	G	E	B	G	GEBG
aridisols	argids	natrargids	durinodic xeric natrargids	G	E	B	H	GEBH
aridisols	argids	natrargids	durinodic natrargids	G	E	B	I	GEBI
aridisols	argids	natrargids	petronodic natrargids	G	E	B	J	GEBJ
aridisols	argids	natrargids	glossic ustic natrargids	G	E	B	K	GEBK
aridisols	argids	natrargids	haplic ustic natrargids	G	E	B	L	GEBL
aridisols	argids	natrargids	haploxeralfic natrargids	G	E	B	M	GEBM
aridisols	argids	natrargids	haplic natrargids	G	E	B	N	GEBN
aridisols	argids	natrargids	vitrixerandic natrargids	G	E	B	O	GEBO
aridisols	argids	natrargids	vitrandic natrargids	G	E	B	P	GEBP
aridisols	argids	natrargids	xeric natrargids	G	E	B	Q	GEBQ
aridisols	argids	natrargids	ustic natrargids	G	E	B	R	GEBR
aridisols	argids	natrargids	glossic natrargids	G	E	B	S	GEBS
aridisols	argids	natrargids	typic natrargids	G	E	B	T	GEBT
aridisols	argids	paleargids	vertic paleargids	G	E	C	A	GECA
aridisols	argids	paleargids	aquic paleargids	G	E	C	B	GECB
aridisols	argids	paleargids	arenic ustic paleargids	G	E	C	C	GECC
aridisols	argids	paleargids	arenic paleargids	G	E	C	D	GECD
aridisols	argids	paleargids	calcic paleargids	G	E	C	E	GECE
aridisols	argids	paleargids	durinodic xeric paleargids	G	E	C	F	GECF
aridisols	argids	paleargids	durinodic paleargids	G	E	C	G	GECG
aridisols	argids	paleargids	petronodic ustic paleargids	G	E	C	H	GECH
aridisols	argids	paleargids	petronodic paleargids	G	E	C	I	GECI
aridisols	argids	paleargids	vitrixerandic paleargids	G	E	C	J	GECJ
aridisols	argids	paleargids	vitrandic paleargids	G	E	C	K	GECK
aridisols	argids	paleargids	xeric paleargids	G	E	C	L	GECL
aridisols	argids	paleargids	ustic paleargids	G	E	C	M	GECM
aridisols	argids	paleargids	typic paleargids	G	E	C	N	GECN
aridisols	argids	gypsiargids	aquic gypsiargids	G	E	D	A	GEDA
aridisols	argids	gypsiargids	durinodic gypsiargids	G	E	D	B	GEDB
aridisols	argids	gypsiargids	vitrixerandic gypsiargids	G	E	D	C	GEDC
aridisols	argids	gypsiargids	vitrandic gypsiargids	G	E	D	D	GEDD
aridisols	argids	gypsiargids	xeric gypsiargids	G	E	D	E	GEDE
aridisols	argids	gypsiargids	ustic gypsiargids	G	E	D	F	GEDF
aridisols	argids	gypsiargids	typic gypsiargids	G	E	D	G	GEDG
aridisols	argids	calciargids	lithic calciargids	G	E	E	A	GEEA
aridisols	argids	calciargids	xerertic calciargids	G	E	E	B	GEEB
aridisols	argids	calciargids	ustertic calciargids	G	E	E	C	GEEC
aridisols	argids	calciargids	vertic calciargids	G	E	E	D	GEED
aridisols	argids	calciargids	aquic calciargids	G	E	E	E	GEEE
aridisols	argids	calciargids	arenic ustic calciargids	G	E	E	F	GEEF
aridisols	argids	calciargids	arenic calciargids	G	E	E	G	GEEG
aridisols	argids	calciargids	durinodic xeric calciargids	G	E	E	H	GEEH
aridisols	argids	calciargids	durinodic calciargids	G	E	E	I	GEEI
aridisols	argids	calciargids	petronodic xeric calciargids	G	E	E	J	GEEJ
aridisols	argids	calciargids	petronodic ustic calciargids	G	E	E	K	GEEK
aridisols	argids	calciargids	petronodic calciargids	G	E	E	L	GEEL
aridisols	argids	calciargids	vitrixerandic calciargids	G	E	E	M	GEEM
aridisols	argids	calciargids	vitrandic calciargids	G	E	E	N	GEEN
aridisols	argids	calciargids	xeric calciargids	G	E	E	O	GEEO

aridisols	argids	calciargids	ustic calciargids	G	E	E	P	GEEP
aridisols	argids	calciargids	typic calciargids	G	E	E	Q	GEEQ
aridisols	argids	haplargids	lithic ruptic-entic haplargids	G	E	F	A	GEFA
aridisols	argids	haplargids	lithic xeric haplargids	G	E	F	B	GEFB
aridisols	argids	haplargids	lithic ustic haplargids	G	E	F	C	GEFC
aridisols	argids	haplargids	lithic haplargids	G	E	F	D	GEFD
aridisols	argids	haplargids	xerertic haplargids	G	E	F	E	GEFE
aridisols	argids	haplargids	ustertic haplargids	G	E	F	F	GEFF
aridisols	argids	haplargids	vertic haplargids	G	E	F	G	GEFG
aridisols	argids	haplargids	aquic haplargids	G	E	F	H	GEFH
aridisols	argids	haplargids	arenic ustic haplargids	G	E	F	I	GEFI
aridisols	argids	haplargids	arenic haplargids	G	E	F	J	GEFJ
aridisols	argids	haplargids	durinodic xeric haplargids	G	E	F	K	GEFK
aridisols	argids	haplargids	durinodic haplargids	G	E	F	L	GEFL
aridisols	argids	haplargids	petronodic ustic haplargids	G	E	F	M	GEFM
aridisols	argids	haplargids	petronodic haplargids	G	E	F	N	GEFN
aridisols	argids	haplargids	vitrixerandic haplargids	G	E	F	O	GEFO
aridisols	argids	haplargids	vitrandic haplargids	G	E	F	P	GEFP
aridisols	argids	haplargids	xeric haplargids	G	E	F	Q	GEFQ
aridisols	argids	haplargids	ustic haplargids	G	E	F	R	GEFR
aridisols	argids	haplargids	typic haplargids	G	E	F	S	GEFS
aridisols	calcids	petrocalcids	aquic petrocalcids	G	F	A	A	GFAA
aridisols	calcids	petrocalcids	natric petrocalcids	G	F	A	B	GFAB
aridisols	calcids	petrocalcids	xeralfic petrocalcids	G	F	A	C	GFAC
aridisols	calcids	petrocalcids	ustalfic petrocalcids	G	F	A	D	GFAD
aridisols	calcids	petrocalcids	argic petrocalcids	G	F	A	E	GFAE
aridisols	calcids	petrocalcids	calcic lithic petrocalcids	G	F	A	F	GFAF
aridisols	calcids	petrocalcids	calcic petrocalcids	G	F	A	G	GFAG
aridisols	calcids	petrocalcids	xeric petrocalcids	G	F	A	H	GFAH
aridisols	calcids	petrocalcids	ustic petrocalcids	G	F	A	I	GFAI
aridisols	calcids	petrocalcids	typic petrocalcids	G	F	A	J	GFAJ
aridisols	calcids	haplocalcids	lithic xeric haplocalcids	G	F	B	A	GFBA
aridisols	calcids	haplocalcids	lithic ustic haplocalcids	G	F	B	B	GFBB
aridisols	calcids	haplocalcids	lithic haplocalcids	G	F	B	C	GFBC
aridisols	calcids	haplocalcids	vertic haplocalcids	G	F	B	D	GFBD
aridisols	calcids	haplocalcids	aquic durinodic haplocalcids	G	F	B	E	GFBE
aridisols	calcids	haplocalcids	aquic haplocalcids	G	F	B	F	GFBF
aridisols	calcids	haplocalcids	duric xeric haplocalcids	G	F	B	G	GFBG
aridisols	calcids	haplocalcids	duric haplocalcids	G	F	B	H	GFBH
aridisols	calcids	haplocalcids	durinodic xeric haplocalcids	G	F	B	I	GFBI
aridisols	calcids	haplocalcids	durinodic haplocalcids	G	F	B	J	GFBJ
aridisols	calcids	haplocalcids	petronodic xeric haplocalcids	G	F	B	K	GFBK
aridisols	calcids	haplocalcids	petronodic ustic haplocalcids	G	F	B	L	GFBL
aridisols	calcids	haplocalcids	petronodic haplocalcids	G	F	B	M	GFBM
aridisols	calcids	haplocalcids	sodic xeric haplocalcids	G	F	B	N	GFBN
aridisols	calcids	haplocalcids	sodic ustic haplocalcids	G	F	B	O	GFBO
aridisols	calcids	haplocalcids	sodic haplocalcids	G	F	B	P	GFBP
aridisols	calcids	haplocalcids	vitrixerandic haplocalcids	G	F	B	Q	GFBQ
aridisols	calcids	haplocalcids	vitrandic haplocalcids	G	F	B	R	GFBR
aridisols	calcids	haplocalcids	xeric haplocalcids	G	F	B	S	GFBS
aridisols	calcids	haplocalcids	ustic haplocalcids	G	F	B	T	GFBT
aridisols	calcids	haplocalcids	typic haplocalcids	G	F	B	U	GFBU
aridisols	cambids	aquicambids	sodic aquicambids	G	G	A	A	GGAA
aridisols	cambids	aquicambids	durinodic xeric aquicambids	G	G	A	B	GGAB
aridisols	cambids	aquicambids	durinodic aquicambids	G	G	A	C	GGAC
aridisols	cambids	aquicambids	petronodic aquicambids	G	G	A	D	GGAD
aridisols	cambids	aquicambids	vitrixerandic aquicambids	G	G	A	E	GGAE
aridisols	cambids	aquicambids	vitrandic aquicambids	G	G	A	F	GGAF
aridisols	cambids	aquicambids	fluventic aquicambids	G	G	A	G	GGAG
aridisols	cambids	aquicambids	xeric aquicambids	G	G	A	H	GGAH

aridisols	cambids	aquicambids	ustic aquicambids	G	G	A	I	GGAI
aridisols	cambids	aquicambids	typic aquicambids	G	G	A	J	GGAJ
aridisols	cambids	petrocambids	sodic petrocambids	G	G	B	A	GGBA
aridisols	cambids	petrocambids	vitrixerandic petrocambids	G	G	B	B	GGBB
aridisols	cambids	petrocambids	vitrandic petrocambids	G	G	B	C	GGBC
aridisols	cambids	petrocambids	xeric petrocambids	G	G	B	D	GGBD
aridisols	cambids	petrocambids	ustic petrocambids	G	G	B	E	GGBE
aridisols	cambids	petrocambids	typic petrocambids	G	G	B	F	GGBF
aridisols	cambids	haplocambids	lithic xeric haplocambids	G	G	C	A	GGCA
aridisols	cambids	haplocambids	lithic ustic haplocambids	G	G	C	B	GGCB
aridisols	cambids	haplocambids	lithic haplocambids	G	G	C	C	GGCC
aridisols	cambids	haplocambids	xerertic haplocambids	G	G	C	D	GGCD
aridisols	cambids	haplocambids	ustertic haplocambids	G	G	C	E	GGCE
aridisols	cambids	haplocambids	vertic haplocambids	G	G	C	F	GGCF
aridisols	cambids	haplocambids	durinodic xeric haplocambids	G	G	C	G	GGCG
aridisols	cambids	haplocambids	durinodic haplocambids	G	G	C	H	GGCH
aridisols	cambids	haplocambids	petronodic xeric haplocambids	G	G	C	I	GGCI
aridisols	cambids	haplocambids	petronodic ustic haplocambids	G	G	C	J	GGCJ
aridisols	cambids	haplocambids	petronodic haplocambids	G	G	C	K	GGCK
aridisols	cambids	haplocambids	sodic xeric haplocambids	G	G	C	L	GGCL
aridisols	cambids	haplocambids	sodic ustic haplocambids	G	G	C	M	GGCM
aridisols	cambids	haplocambids	sodic haplocambids	G	G	C	N	GGCN
aridisols	cambids	haplocambids	vitrixerandic haplocambids	G	G	C	O	GGCO
aridisols	cambids	haplocambids	vitrandic haplocambids	G	G	C	P	GGCP
aridisols	cambids	haplocambids	xerofluventic haplocambids	G	G	C	Q	GGCQ
aridisols	cambids	haplocambids	ustifluventic haplocambids	G	G	C	R	GGCR
aridisols	cambids	haplocambids	fluventic haplocambids	G	G	C	S	GGCS
aridisols	cambids	haplocambids	anthropic haplocambids	G	G	C	T	GGCT
aridisols	cambids	haplocambids	xeric haplocambids	G	G	C	U	GGCU
aridisols	cambids	haplocambids	ustic haplocambids	G	G	C	V	GGCV
aridisols	cambids	haplocambids	typic haplocambids	G	G	C	W	GGCW
ultisols	aquults	plinthaquults	kandic plinthaquults	H	A	A	A	HAAA
ultisols	aquults	plinthaquults	typic plinthaquults	H	A	A	B	HAAB
ultisols	aquults	fragiaquults	aeric fragiaquults	H	A	B	A	HABA
ultisols	aquults	fragiaquults	plinthic fragiaquults	H	A	B	B	HABB
ultisols	aquults	fragiaquults	umbric fragiaquults	H	A	B	C	HABC
ultisols	aquults	fragiaquults	typic fragiaquults	H	A	B	D	HABD
ultisols	aquults	albaquults	vertic albaquults	H	A	C	A	HACA
ultisols	aquults	albaquults	kandic albaquults	H	A	C	B	HACB
ultisols	aquults	albaquults	aeric albaquults	H	A	C	C	HACC
ultisols	aquults	albaquults	typic albaquults	H	A	C	D	HACD
ultisols	aquults	kandiaquults	acraquoxic kandiaquults	H	A	D	A	HADA
ultisols	aquults	kandiaquults	arenic plinthic kandiaquults	H	A	D	B	HADB
ultisols	aquults	kandiaquults	arenic umbric kandiaquults	H	A	D	C	HADC
ultisols	aquults	kandiaquults	arenic kandiaquults	H	A	D	D	HADD
ultisols	aquults	kandiaquults	grossarenic kandiaquults	H	A	D	E	HADE
ultisols	aquults	kandiaquults	plinthic kandiaquults	H	A	D	F	HADF
ultisols	aquults	kandiaquults	aeric kandiaquults	H	A	D	G	HADG
ultisols	aquults	kandiaquults	umbric kandiaquults	H	A	D	H	HADH
ultisols	aquults	kandiaquults	typic kandiaquults	H	A	D	I	HADI
ultisols	aquults	kanhaplaquults	aquandic kanhaplaquults	H	A	E	A	HAEA
ultisols	aquults	kanhaplaquults	plinthic kanhaplaquults	H	A	E	B	HAEB
ultisols	aquults	kanhaplaquults	aeric umbric kanhaplaquults	H	A	E	C	HAEC
ultisols	aquults	kanhaplaquults	aeric kanhaplaquults	H	A	E	D	HAED
ultisols	aquults	kanhaplaquults	umbric kanhaplaquults	H	A	E	E	HAEE
ultisols	aquults	kanhaplaquults	typic kanhaplaquults	H	A	E	F	HAEF
ultisols	aquults	paleaquults	vertic paleaquults	H	A	F	A	HAFA
ultisols	aquults	paleaquults	arenic plinthic paleaquults	H	A	F	B	HAFB
ultisols	aquults	paleaquults	arenic umbric paleaquults	H	A	F	C	HAFC
ultisols	aquults	paleaquults	arenic paleaquults	H	A	F	D	HAFD

ultisols	aquults	paleaquults	grossarenic paleaquults	H	A	F	E	HAFE
ultisols	aquults	paleaquults	plinthic paleaquults	H	A	F	F	HAFF
ultisols	aquults	paleaquults	aeric paleaquults	H	A	F	G	HAFG
ultisols	aquults	paleaquults	umbric paleaquults	H	A	F	H	HAFH
ultisols	aquults	paleaquults	typic paleaquults	H	A	F	I	HAFI
ultisols	aquults	umbraquults	plinthic umbraquults	H	A	G	A	HAGA
ultisols	aquults	umbraquults	typic umbraquults	H	A	G	B	HAGB
ultisols	aquults	epiaquults	vertic epiaquults	H	A	H	A	HAHA
ultisols	aquults	epiaquults	aeric fragic epiaquults	H	A	H	B	HAHB
ultisols	aquults	epiaquults	arenic epiaquults	H	A	H	C	HAHC
ultisols	aquults	epiaquults	grossarenic epiaquults	H	A	H	D	HAHD
ultisols	aquults	epiaquults	fragic epiaquults	H	A	H	E	HAHE
ultisols	aquults	epiaquults	aeric epiaquults	H	A	H	F	HAHF
ultisols	aquults	epiaquults	typic epiaquults	H	A	H	G	HAHG
ultisols	aquults	endoaquults	arenic endoaquults	H	A	I	A	HAIA
ultisols	aquults	endoaquults	grossarenic endoaquults	H	A	I	B	HAIB
ultisols	aquults	endoaquults	aeric endoaquults	H	A	I	C	HAIC
ultisols	aquults	endoaquults	typic endoaquults	H	A	I	D	HAID
ultisols	humults	sombrihumults	typic sombrihumults	H	B	A	A	HBAA
ultisols	humults	plinthohumults	typic plinthohumults	H	B	B	A	HBBA
ultisols	humults	kandihumults	andic ombroaquic kandihumults	H	B	C	A	HBCA
ultisols	humults	kandihumults	ustandic kandihumults	H	B	C	B	HBCB
ultisols	humults	kandihumults	andic kandihumults	H	B	C	C	HBCC
ultisols	humults	kandihumults	aquic kandihumults	H	B	C	D	HBCD
ultisols	humults	kandihumults	ombroaquic kandihumults	H	B	C	E	HBCE
ultisols	humults	kandihumults	plinthic kandihumults	H	B	C	F	HBCF
ultisols	humults	kandihumults	ustic kandihumults	H	B	C	G	HBCG
ultisols	humults	kandihumults	xeric kandihumults	H	B	C	H	HBCH
ultisols	humults	kandihumults	anthropic kandihumults	H	B	C	I	HBCI
ultisols	humults	kandihumults	typic kandihumults	H	B	C	J	HBCJ
ultisols	humults	kanhaplohumults	lithic kanhaplohumults	H	B	D	A	HBDA
ultisols	humults	kanhaplohumults	ustandic kanhaplohumults	H	B	D	B	HBDB
ultisols	humults	kanhaplohumults	andic kanhaplohumults	H	B	D	C	HBDC
ultisols	humults	kanhaplohumults	aquic kanhaplohumults	H	B	D	D	HBDD
ultisols	humults	kanhaplohumults	ombroaquic kanhaplohumults	H	B	D	E	HBDE
ultisols	humults	kanhaplohumults	ustic kanhaplohumults	H	B	D	F	HBDF
ultisols	humults	kanhaplohumults	xeric kanhaplohumults	H	B	D	G	HBDG
ultisols	humults	kanhaplohumults	anthropic kanhaplohumults	H	B	D	H	HBDH
ultisols	humults	kanhaplohumults	typic kanhaplohumults	H	B	D	I	HBDI
ultisols	humults	palehumults	aquandic palehumults	H	B	E	A	HBEA
ultisols	humults	palehumults	andic palehumults	H	B	E	B	HBEB
ultisols	humults	palehumults	aquic palehumults	H	B	E	C	HBEC
ultisols	humults	palehumults	plinthic palehumults	H	B	E	D	HBED
ultisols	humults	palehumults	oxyaquic palehumults	H	B	E	E	HBEE
ultisols	humults	palehumults	ustic palehumults	H	B	E	F	HBEF
ultisols	humults	palehumults	xeric palehumults	H	B	E	G	HBEG
ultisols	humults	palehumults	typic palehumults	H	B	E	H	HBEH
ultisols	humults	haplohumults	lithic haplohumults	H	B	F	A	HBFA
ultisols	humults	haplohumults	aquandic haplohumults	H	B	F	B	HBFB
ultisols	humults	haplohumults	aquic haplohumults	H	B	F	C	HBFC
ultisols	humults	haplohumults	andic haplohumults	H	B	F	D	HBFD
ultisols	humults	haplohumults	plinthic haplohumults	H	B	F	E	HBFE
ultisols	humults	haplohumults	oxyaquic haplohumults	H	B	F	F	HBFF
ultisols	humults	haplohumults	ustic haplohumults	H	B	F	G	HBFG
ultisols	humults	haplohumults	xeric haplohumults	H	B	F	H	HBFH
ultisols	humults	haplohumults	typic haplohumults	H	B	F	I	HBFI
ultisols	udults	plinthudults	typic plinthudults	H	C	A	A	HCAA
ultisols	udults	fragiudults	arenic fragiudults	H	C	B	A	HCBA
ultisols	udults	fragiudults	plinthaquic fragiudults	H	C	B	B	HCBB
ultisols	udults	fragiudults	glossaquic fragiudults	H	C	B	C	HCBC

ultisols	udults	fragiudults	aquic fragiudults	H	C	B	D	HCBD
ultisols	udults	fragiudults	plinthic fragiudults	H	C	B	E	HCBE
ultisols	udults	fragiudults	glossic fragiudults	H	C	B	F	HCBF
ultisols	udults	fragiudults	humic fragiudults	H	C	B	G	HCBG
ultisols	udults	fragiudults	typic fragiudults	H	C	B	H	HCBH
ultisols	udults	kandiudults	arenic plinthaquic kandiudults	H	C	C	A	HCCA
ultisols	udults	kandiudults	aquic arenic kandiudults	H	C	C	B	HCCB
ultisols	udults	kandiudults	arenic plinthic kandiudults	H	C	C	C	HCCC
ultisols	udults	kandiudults	arenic rhodic kandiudults	H	C	C	D	HCCD
ultisols	udults	kandiudults	arenic kandiudults	H	C	C	E	HCCE
ultisols	udults	kandiudults	grossarenic plinthic kandiudults	H	C	C	F	HCCF
ultisols	udults	kandiudults	grossarenic kandiudults	H	C	C	G	HCCG
ultisols	udults	kandiudults	acrudoxic plinthic kandiudults	H	C	C	H	HCCH
ultisols	udults	kandiudults	acrudoxic kandiudults	H	C	C	I	HCCI
ultisols	udults	kandiudults	plinthaquic kandiudults	H	C	C	J	HCCJ
ultisols	udults	kandiudults	aquandic kandiudults	H	C	C	K	HCCK
ultisols	udults	kandiudults	andic kandiudults	H	C	C	L	HCCL
ultisols	udults	kandiudults	aquic kandiudults	H	C	C	M	HCCM
ultisols	udults	kandiudults	plinthic kandiudults	H	C	C	N	HCCN
ultisols	udults	kandiudults	ombroaquic kandiudults	H	C	C	O	HCCO
ultisols	udults	kandiudults	oxyaquic kandiudults	H	C	C	P	HCCP
ultisols	udults	kandiudults	sombric kandiudults	H	C	C	Q	HCCQ
ultisols	udults	kandiudults	rhodic kandiudults	H	C	C	R	HCCR
ultisols	udults	kandiudults	typic kandiudults	H	C	C	S	HCCS
ultisols	udults	kanhapludults	lithic kanhapludults	H	C	D	A	HCDA
ultisols	udults	kanhapludults	plinthaquic kanhapludults	H	C	D	B	HCDB
ultisols	udults	kanhapludults	arenic plinthic kanhapludults	H	C	D	C	HCDC
ultisols	udults	kanhapludults	arenic kanhapludults	H	C	D	D	HCDD
ultisols	udults	kanhapludults	acrudoxic kanhapludults	H	C	D	E	HCDE
ultisols	udults	kanhapludults	fragiaquic kanhapludults	H	C	D	F	HCDF
ultisols	udults	kanhapludults	andic kanhapludults	H	C	D	G	HCDG
ultisols	udults	kanhapludults	aquic kanhapludults	H	C	D	H	HCDH
ultisols	udults	kanhapludults	ombroaquic kanhapludults	H	C	D	I	HCDI
ultisols	udults	kanhapludults	oxyaquic kanhapludults	H	C	D	J	HCDJ
ultisols	udults	kanhapludults	plinthic kanhapludults	H	C	D	K	HCDK
ultisols	udults	kanhapludults	fragic kanhapludults	H	C	D	L	HCDL
ultisols	udults	kanhapludults	rhodic kanhapludults	H	C	D	M	HCDM
ultisols	udults	kanhapludults	typic kanhapludults	H	C	D	N	HCDN
ultisols	udults	paleudults	vertic paleudults	H	C	E	A	HCEA
ultisols	udults	paleudults	spodic paleudults	H	C	E	B	HCEB
ultisols	udults	paleudults	arenic plinthaquic paleudults	H	C	E	C	HCEC
ultisols	udults	paleudults	aquic arenic paleudults	H	C	E	D	HCED
ultisols	udults	paleudults	anthraquic paleudults	H	C	E	E	HCEE
ultisols	udults	paleudults	plinthaquic paleudults	H	C	E	F	HCEF
ultisols	udults	paleudults	fragiaquic paleudults	H	C	E	G	HCEG
ultisols	udults	paleudults	aquic paleudults	H	C	E	H	HCEH
ultisols	udults	paleudults	oxyaquic paleudults	H	C	E	I	HCEI
ultisols	udults	paleudults	lamellic paleudults	H	C	E	J	HCEJ
ultisols	udults	paleudults	arenic plinthic paleudults	H	C	E	K	HCEK
ultisols	udults	paleudults	psammentic paleudults	H	C	E	L	HCEL
ultisols	udults	paleudults	grossarenic plinthic paleudults	H	C	E	M	HCEM
ultisols	udults	paleudults	plinthic paleudults	H	C	E	N	HCEN
ultisols	udults	paleudults	arenic rhodic paleudults	H	C	E	O	HCEO
ultisols	udults	paleudults	arenic paleudults	H	C	E	P	HCEP
ultisols	udults	paleudults	grossarenic paleudults	H	C	E	Q	HCEQ
ultisols	udults	paleudults	fragic paleudults	H	C	E	R	HCER
ultisols	udults	paleudults	rhodic paleudults	H	C	E	S	HCES
ultisols	udults	paleudults	typic paleudults	H	C	E	T	HCET
ultisols	udults	rhodudults	lithic rhodudults	H	C	F	A	HCFA
ultisols	udults	rhodudults	psammentic rhodudults	H	C	F	B	HCFB

ultisols	udults	rhodudults	typic rhodudults	H	C	F	C	HCFC
ultisols	udults	hapludults	lithic-ruptic-entic hapludults	H	C	G	A	HCGA
ultisols	udults	hapludults	lithic hapludults	H	C	G	B	HCGB
ultisols	udults	hapludults	vertic hapludults	H	C	G	C	HCGC
ultisols	udults	hapludults	fragiaquic hapludults	H	C	G	D	HCGD
ultisols	udults	hapludults	aquic arenic hapludults	H	C	G	E	HCGE
ultisols	udults	hapludults	aquic hapludults	H	C	G	F	HCGF
ultisols	udults	hapludults	fragic hapludults	H	C	G	G	HCGG
ultisols	udults	hapludults	oxyaquic hapludults	H	C	G	H	HCGH
ultisols	udults	hapludults	lamellic hapludults	H	C	G	I	HCGI
ultisols	udults	hapludults	psammentic hapludults	H	C	G	J	HCGJ
ultisols	udults	hapludults	arenic hapludults	H	C	G	K	HCGK
ultisols	udults	hapludults	grossarenic hapludults	H	C	G	L	HCGL
ultisols	udults	hapludults	inceptic hapludults	H	C	G	M	HCGM
ultisols	udults	hapludults	humic hapludults	H	C	G	N	HCGN
ultisols	udults	hapludults	typic hapludults	H	C	G	O	HCGO
ultisols	ustults	plinthustults	haplic plinthustults	H	D	A	A	HDAA
ultisols	ustults	plinthustults	typic plinthustults	H	D	A	B	HDAB
ultisols	ustults	kandiustults	acrustoxic kandiustults	H	D	B	A	HDBA
ultisols	ustults	kandiustults	aquic kandiustults	H	D	B	B	HDBB
ultisols	ustults	kandiustults	arenic plinthic kandiustults	H	D	B	C	HDBC
ultisols	ustults	kandiustults	arenic kandiustults	H	D	B	D	HDBD
ultisols	ustults	kandiustults	udandic kandiustults	H	D	B	E	HDBE
ultisols	ustults	kandiustults	andic kandiustults	H	D	B	F	HDBF
ultisols	ustults	kandiustults	plinthic kandiustults	H	D	B	G	HDBG
ultisols	ustults	kandiustults	aridic kandiustults	H	D	B	H	HDBH
ultisols	ustults	kandiustults	udic kandiustults	H	D	B	I	HDBI
ultisols	ustults	kandiustults	rhodic kandiustults	H	D	B	J	HDBJ
ultisols	ustults	kandiustults	typic kandiustults	H	D	B	K	HDBK
ultisols	ustults	kanhaplustults	lithic kanhaplustults	H	D	C	A	HDCA
ultisols	ustults	kanhaplustults	acrustoxic kanhaplustults	H	D	C	B	HDCB
ultisols	ustults	kanhaplustults	aquic kanhaplustults	H	D	C	C	HDCC
ultisols	ustults	kanhaplustults	arenic kanhaplustults	H	D	C	D	HDCD
ultisols	ustults	kanhaplustults	udandic kanhaplustults	H	D	C	E	HDCE
ultisols	ustults	kanhaplustults	andic kanhaplustults	H	D	C	F	HDCF
ultisols	ustults	kanhaplustults	plinthic kanhaplustults	H	D	C	G	HDCG
ultisols	ustults	kanhaplustults	ombroaquic kanhaplustults	H	D	C	H	HDCH
ultisols	ustults	kanhaplustults	aridic kanhaplustults	H	D	C	I	HDCI
ultisols	ustults	kanhaplustults	udic kanhaplustults	H	D	C	J	HDCJ
ultisols	ustults	kanhaplustults	rhodic kanhaplustults	H	D	C	K	HDCK
ultisols	ustults	kanhaplustults	typic kanhaplustults	H	D	C	L	HDCL
ultisols	ustults	paleustults	typic paleustults	H	D	D	A	HDDA
ultisols	ustults	rhodustults	lithic rhodustults	H	D	E	A	HDEA
ultisols	ustults	rhodustults	psammentic rhodustults	H	D	E	B	HDEB
ultisols	ustults	rhodustults	typic rhodustults	H	D	E	C	HDEC
ultisols	ustults	haplustults	lithic haplustults	H	D	F	A	HDFA
ultisols	ustults	haplustults	petroferric haplustults	H	D	F	B	HDFB
ultisols	ustults	haplustults	aquic haplustults	H	D	F	C	HDFC
ultisols	ustults	haplustults	arenic haplustults	H	D	F	D	HDFD
ultisols	ustults	haplustults	ombroaquic haplustults	H	D	F	E	HDFE
ultisols	ustults	haplustults	plinthic haplustults	H	D	F	F	HDFF
ultisols	ustults	haplustults	kanhaplic haplustults	H	D	F	G	HDFG
ultisols	ustults	haplustults	typic haplustults	H	D	F	H	HDFH
ultisols	xerults	palexerults	aquandic palexerults	H	E	A	A	HEAA
ultisols	xerults	palexerults	aquic palexerults	H	E	A	B	HEAB
ultisols	xerults	palexerults	andic palexerults	H	E	A	C	HEAC
ultisols	xerults	palexerults	typic palexerults	H	E	A	D	HEAD
ultisols	xerults	haploxerults	lithic ruptic-inceptic haploxerults	H	E	B	A	HEBA
ultisols	xerults	haploxerults	lithic haploxerults	H	E	B	B	HEBB
ultisols	xerults	haploxerults	aquic haploxerults	H	E	B	C	HEBC

ultisols	xerults	haploxerults	andic haploxerults	H	E	B	D	HEBD
ultisols	xerults	haploxerults	lamellic haploxerults	H	E	B	E	HEBE
ultisols	xerults	haploxerults	psammentic haploxerults	H	E	B	F	HEBF
ultisols	xerults	haploxerults	arenic haploxerults	H	E	B	G	HEBG
ultisols	xerults	haploxerults	grossarenic haploxerults	H	E	B	H	HEBH
ultisols	xerults	haploxerults	typic haploxerults	H	E	B	I	HEBI
mollisols	albolls	natralbolls	leptic natralbolls	I	A	A	A	IAAA
mollisols	albolls	natralbolls	typic natralbolls	I	A	A	B	IAAB
mollisols	albolls	argialbolls	xerertic argialbolls	I	A	B	A	IABA
mollisols	albolls	argialbolls	vertic argialbolls	I	A	B	B	IABB
mollisols	albolls	argialbolls	argiaquic xeric argialbolls	I	A	B	C	IABC
mollisols	albolls	argialbolls	argiaquic argialbolls	I	A	B	D	IABD
mollisols	albolls	argialbolls	xeric argialbolls	I	A	B	E	IABE
mollisols	albolls	argialbolls	aquandic argialbolls	I	A	B	F	IABF
mollisols	albolls	argialbolls	typic argialbolls	I	A	B	G	IABG
mollisols	aquolls	cryaquolls	vertic cryaquolls	I	B	A	A	IBAA
mollisols	aquolls	cryaquolls	histic cryaquolls	I	B	A	B	IBAB
mollisols	aquolls	cryaquolls	thapto-histic cryaquolls	I	B	A	C	IBAC
mollisols	aquolls	cryaquolls	aquandic cryaquolls	I	B	A	D	IBAD
mollisols	aquolls	cryaquolls	argic cryaquolls	I	B	A	E	IBAE
mollisols	aquolls	cryaquolls	calcic cryaquolls	I	B	A	F	IBAF
mollisols	aquolls	cryaquolls	cumulic cryaquolls	I	B	A	G	IBAG
mollisols	aquolls	cryaquolls	typic cryaquolls	I	B	A	H	IBAH
mollisols	aquolls	duraquolls	natric duraquolls	I	B	B	A	IBBA
mollisols	aquolls	duraquolls	vertic duraquolls	I	B	B	B	IBBB
mollisols	aquolls	duraquolls	argic duraquolls	I	B	B	C	IBBC
mollisols	aquolls	duraquolls	typic duraquolls	I	B	B	D	IBBD
mollisols	aquolls	natraquolls	petrocalcic natraquolls	I	B	C	A	IBCA
mollisols	aquolls	natraquolls	vertic natraquolls	I	B	C	B	IBCB
mollisols	aquolls	natraquolls	glossic natraquolls	I	B	C	C	IBCC
mollisols	aquolls	natraquolls	typic natraquolls	I	B	C	D	IBCD
mollisols	aquolls	calciaquolls	petrocalcic calciaquolls	I	B	D	A	IBDA
mollisols	aquolls	calciaquolls	aeric calciaquolls	I	B	D	B	IBDB
mollisols	aquolls	calciaquolls	typic calciaquolls	I	B	D	C	IBDC
mollisols	aquolls	argiaquolls	arenic argiaquolls	I	B	E	A	IBEA
mollisols	aquolls	argiaquolls	grossarenic argiaquolls	I	B	E	B	IBEB
mollisols	aquolls	argiaquolls	vertic argiaquolls	I	B	E	C	IBEC
mollisols	aquolls	argiaquolls	abruptic argiaquolls	I	B	E	D	IBED
mollisols	aquolls	argiaquolls	typic argiaquolls	I	B	E	E	IBEE
mollisols	aquolls	epiaquolls	cumulic vertic epiaquolls	I	B	F	A	IBFA
mollisols	aquolls	epiaquolls	fluvaquentic vertic epiaquolls	I	B	F	B	IBFB
mollisols	aquolls	epiaquolls	vertic epiaquolls	I	B	F	C	IBFC
mollisols	aquolls	epiaquolls	histic epiaquolls	I	B	F	D	IBFD
mollisols	aquolls	epiaquolls	thapto-histic epiaquolls	I	B	F	E	IBFE
mollisols	aquolls	epiaquolls	aquandic epiaquolls	I	B	F	F	IBFF
mollisols	aquolls	epiaquolls	duric epiaquolls	I	B	F	G	IBFG
mollisols	aquolls	epiaquolls	cumulic epiaquolls	I	B	F	H	IBFH
mollisols	aquolls	epiaquolls	fluvaquentic epiaquolls	I	B	F	I	IBFI
mollisols	aquolls	epiaquolls	typic epiaquolls	I	B	F	J	IBFJ
mollisols	aquolls	endoaquolls	lithic endoaquolls	I	B	G	A	IBGA
mollisols	aquolls	endoaquolls	cumulic vertic endoaquolls	I	B	G	B	IBGB
mollisols	aquolls	endoaquolls	fluvaquentic vertic endoaquolls	I	B	G	C	IBGC
mollisols	aquolls	endoaquolls	vertic endoaquolls	I	B	G	D	IBGD
mollisols	aquolls	endoaquolls	histic endoaquolls	I	B	G	E	IBGE
mollisols	aquolls	endoaquolls	thapto-histic endoaquolls	I	B	G	F	IBGF
mollisols	aquolls	endoaquolls	aquandic endoaquolls	I	B	G	G	IBGG
mollisols	aquolls	endoaquolls	duric endoaquolls	I	B	G	H	IBGH
mollisols	aquolls	endoaquolls	cumulic endoaquolls	I	B	G	I	IBGI
mollisols	aquolls	endoaquolls	fluvaquentic endoaquolls	I	B	G	J	IBGJ
mollisols	aquolls	endoaquolls	typic endoaquolls	I	B	G	K	IBGK

mollisols	rendolls	cryrendolls	lithic cryrendolls	I	C	A	A	ICAA
mollisols	rendolls	cryrendolls	typic cryrendolls	I	C	A	B	ICAB
mollisols	rendolls	haprendolls	lithic haprendolls	I	C	B	A	ICBA
mollisols	rendolls	haprendolls	vertic haprendolls	I	C	B	B	ICBB
mollisols	rendolls	haprendolls	inceptic haprendolls	I	C	B	C	ICBC
mollisols	rendolls	haprendolls	entic haprendolls	I	C	B	D	ICBD
mollisols	rendolls	haprendolls	typic haprendolls	I	C	B	E	ICBE
mollisols	gelolls	haplogelolls	lithic haplogelolls	I	D	A	A	IDAA
mollisols	gelolls	haplogelolls	andic haplogelolls	I	D	A	B	IDAB
mollisols	gelolls	haplogelolls	aquic haplogelolls	I	D	A	C	IDAC
mollisols	gelolls	haplogelolls	oxyaquic haplogelolls	I	D	A	D	IDAD
mollisols	gelolls	haplogelolls	turbic haplogelolls	I	D	A	E	IDAE
mollisols	gelolls	haplogelolls	cumulic haplogelolls	I	D	A	F	IDAF
mollisols	gelolls	haplogelolls	typic haplogelolls	I	D	A	G	IDAG
mollisols	cryolls	duricryolls	argic duricryolls	I	E	A	A	IEAA
mollisols	cryolls	duricryolls	calcic duricryolls	I	E	A	B	IEAB
mollisols	cryolls	duricryolls	typic duricryolls	I	E	A	C	IEAC
mollisols	cryolls	natricryolls	typic natricryolls	I	E	B	A	IEBA
mollisols	cryolls	palecryolls	aquic palecryolls	I	E	C	A	IECA
mollisols	cryolls	palecryolls	oxyaquic palecryolls	I	E	C	B	IECB
mollisols	cryolls	palecryolls	abruptic palecryolls	I	E	C	C	IECC
mollisols	cryolls	palecryolls	pachic palecryolls	I	E	C	D	IECD
mollisols	cryolls	palecryolls	ustic palecryolls	I	E	C	E	IECE
mollisols	cryolls	palecryolls	xeric palecryolls	I	E	C	F	IECF
mollisols	cryolls	palecryolls	typic palecryolls	I	E	C	G	IECG
mollisols	cryolls	argicryolls	lithic argicryolls	I	E	D	A	IEDA
mollisols	cryolls	argicryolls	vertic argicryolls	I	E	D	B	IEDB
mollisols	cryolls	argicryolls	andic argicryolls	I	E	D	C	IEDC
mollisols	cryolls	argicryolls	vitrandic argicryolls	I	E	D	D	IEDD
mollisols	cryolls	argicryolls	abruptic argicryolls	I	E	D	E	IEDE
mollisols	cryolls	argicryolls	aquic argicryolls	I	E	D	F	IEDF
mollisols	cryolls	argicryolls	oxyaquic argicryolls	I	E	D	G	IEDG
mollisols	cryolls	argicryolls	calcic pachic argicryolls	I	E	D	H	IEDH
mollisols	cryolls	argicryolls	pachic argicryolls	I	E	D	I	IEDI
mollisols	cryolls	argicryolls	calcic argicryolls	I	E	D	J	IEDJ
mollisols	cryolls	argicryolls	alfic argicryolls	I	E	D	K	IEDK
mollisols	cryolls	argicryolls	ustic argicryolls	I	E	D	L	IEDL
mollisols	cryolls	argicryolls	xeric argicryolls	I	E	D	M	IEDM
mollisols	cryolls	argicryolls	typic argicryolls	I	E	D	N	IEDN
mollisols	cryolls	calcicryolls	lithic calcicryolls	I	E	E	A	IEEA
mollisols	cryolls	calcicryolls	vitrandic calcicryolls	I	E	E	B	IEEB
mollisols	cryolls	calcicryolls	petrocalcic calcicryolls	I	E	E	C	IEEC
mollisols	cryolls	calcicryolls	pachic calcicryolls	I	E	E	D	IEED
mollisols	cryolls	calcicryolls	ustic calcicryolls	I	E	E	E	IEEE
mollisols	cryolls	calcicryolls	xeric calcicryolls	I	E	E	F	IEEF
mollisols	cryolls	calcicryolls	typic calcicryolls	I	E	E	G	IEEG
mollisols	cryolls	haplocryolls	lithic haplocryolls	I	E	F	A	IEFA
mollisols	cryolls	haplocryolls	vertic haplocryolls	I	E	F	B	IEFB
mollisols	cryolls	haplocryolls	andic haplocryolls	I	E	F	C	IEFC
mollisols	cryolls	haplocryolls	vitrandic haplocryolls	I	E	F	D	IEFD
mollisols	cryolls	haplocryolls	aquic cumulic haplocryolls	I	E	F	E	IEFE
mollisols	cryolls	haplocryolls	cumulic haplocryolls	I	E	F	F	IEFF
mollisols	cryolls	haplocryolls	fluvaquentic haplocryolls	I	E	F	G	IEFG
mollisols	cryolls	haplocryolls	aquic haplocryolls	I	E	F	H	IEFH
mollisols	cryolls	haplocryolls	oxyaquic haplocryolls	I	E	F	I	IEFI
mollisols	cryolls	haplocryolls	calcic pachic haplocryolls	I	E	F	J	IEFJ
mollisols	cryolls	haplocryolls	pachic haplocryolls	I	E	F	K	IEFK
mollisols	cryolls	haplocryolls	fluventic haplocryolls	I	E	F	L	IEFL
mollisols	cryolls	haplocryolls	calcic haplocryolls	I	E	F	M	IEFM
mollisols	cryolls	haplocryolls	ustic haplocryolls	I	E	F	N	IEFN

mollisols	cryolls	haplocryolls	xeric haplocryolls	I	E	F	O	IEFO
mollisols	cryolls	haplocryolls	typic haplocryolls	I	E	F	P	IEFP
mollisols	xerolls	durixerolls	vertic durixerolls	I	F	A	A	IFAA
mollisols	xerolls	durixerolls	vitritorrandic durixerolls	I	F	A	B	IFAB
mollisols	xerolls	durixerolls	vitrandic durixerolls	I	F	A	C	IFAC
mollisols	xerolls	durixerolls	aquic durixerolls	I	F	A	D	IFAD
mollisols	xerolls	durixerolls	paleargidic durixerolls	I	F	A	E	IFAE
mollisols	xerolls	durixerolls	abruptic argiduridic durixerolls	I	F	A	F	IFAF
mollisols	xerolls	durixerolls	cambidic durixerolls	I	F	A	G	IFAG
mollisols	xerolls	durixerolls	haploduridic durixerolls	I	F	A	H	IFAH
mollisols	xerolls	durixerolls	argidic durixerolls	I	F	A	I	IFAI
mollisols	xerolls	durixerolls	argiduridic durixerolls	I	F	A	J	IFAJ
mollisols	xerolls	durixerolls	haplic palexerollic durixerolls	I	F	A	K	IFAK
mollisols	xerolls	durixerolls	palexerollic durixerolls	I	F	A	L	IFAL
mollisols	xerolls	durixerolls	haplic haploxerollic durixerolls	I	F	A	M	IFAM
mollisols	xerolls	durixerolls	haploxerollic durixerolls	I	F	A	N	IFAN
mollisols	xerolls	durixerolls	haplic durixerolls	I	F	A	O	IFAO
mollisols	xerolls	durixerolls	typic durixerolls	I	F	A	P	IFAP
mollisols	xerolls	natrixerolls	vertic natrixerolls	I	F	B	A	IFBA
mollisols	xerolls	natrixerolls	aquic duric natrixerolls	I	F	B	B	IFBB
mollisols	xerolls	natrixerolls	aquic natrixerolls	I	F	B	C	IFBC
mollisols	xerolls	natrixerolls	aridic natrixerolls	I	F	B	D	IFBD
mollisols	xerolls	natrixerolls	duric natrixerolls	I	F	B	E	IFBE
mollisols	xerolls	natrixerolls	typic natrixerolls	I	F	B	F	IFBF
mollisols	xerolls	palexerolls	vertic palexerolls	I	F	C	A	IFCA
mollisols	xerolls	palexerolls	vitrandic palexerolls	I	F	C	B	IFCB
mollisols	xerolls	palexerolls	aquic palexerolls	I	F	C	C	IFCC
mollisols	xerolls	palexerolls	pachic palexerolls	I	F	C	D	IFCD
mollisols	xerolls	palexerolls	petrocalcidic palexerolls	I	F	C	E	IFCE
mollisols	xerolls	palexerolls	duric palexerolls	I	F	C	F	IFCF
mollisols	xerolls	palexerolls	aridic palexerolls	I	F	C	G	IFCG
mollisols	xerolls	palexerolls	petrocalcic palexerolls	I	F	C	H	IFCH
mollisols	xerolls	palexerolls	ultic palexerolls	I	F	C	I	IFCI
mollisols	xerolls	palexerolls	haplic palexerolls	I	F	C	J	IFCJ
mollisols	xerolls	palexerolls	typic palexerolls	I	F	C	K	IFCK
mollisols	xerolls	calcixerolls	aridic lithic calcixerolls	I	F	D	A	IFDA
mollisols	xerolls	calcixerolls	lithic calcixerolls	I	F	D	B	IFDB
mollisols	xerolls	calcixerolls	vertic calcixerolls	I	F	D	C	IFDC
mollisols	xerolls	calcixerolls	aquic calcixerolls	I	F	D	D	IFDD
mollisols	xerolls	calcixerolls	oxyaquic calcixerolls	I	F	D	E	IFDE
mollisols	xerolls	calcixerolls	pachic calcixerolls	I	F	D	F	IFDF
mollisols	xerolls	calcixerolls	vitrandic calcixerolls	I	F	D	G	IFDG
mollisols	xerolls	calcixerolls	aridic calcixerolls	I	F	D	H	IFDH
mollisols	xerolls	calcixerolls	vermic calcixerolls	I	F	D	I	IFDI
mollisols	xerolls	calcixerolls	typic calcixerolls	I	F	D	J	IFDJ
mollisols	xerolls	argixerolls	aridic lithic argixerolls	I	F	E	A	IFEA
mollisols	xerolls	argixerolls	lithic ultic argixerolls	I	F	E	B	IFEB
mollisols	xerolls	argixerolls	lithic argixerolls	I	F	E	C	IFEC
mollisols	xerolls	argixerolls	torrertic argixerolls	I	F	E	D	IFED
mollisols	xerolls	argixerolls	vertic argixerolls	I	F	E	E	IFEE
mollisols	xerolls	argixerolls	andic argixerolls	I	F	E	F	IFEF
mollisols	xerolls	argixerolls	vitritorrandic argixerolls	I	F	E	G	IFEG
mollisols	xerolls	argixerolls	vitrandic argixerolls	I	F	E	H	IFEH
mollisols	xerolls	argixerolls	aquultic argixerolls	I	F	E	I	IFEI
mollisols	xerolls	argixerolls	aquic argixerolls	I	F	E	J	IFEJ
mollisols	xerolls	argixerolls	oxyaquic argixerolls	I	F	E	K	IFEK
mollisols	xerolls	argixerolls	alfic argixerolls	I	F	E	L	IFEL
mollisols	xerolls	argixerolls	calcic pachic argixerolls	I	F	E	M	IFEM
mollisols	xerolls	argixerolls	pachic ultic argixerolls	I	F	E	N	IFEN
mollisols	xerolls	argixerolls	pachic argixerolls	I	F	E	O	IFEO

mollisols	xerolls	argixerolls	argiduridic argixerolls	I	F	E	P	IFEP
mollisols	xerolls	argixerolls	duric argixerolls	I	F	E	Q	IFEQ
mollisols	xerolls	argixerolls	calciargidic argixerolls	I	F	E	R	IFER
mollisols	xerolls	argixerolls	aridic argixerolls	I	F	E	S	IFES
mollisols	xerolls	argixerolls	calcic argixerolls	I	F	E	T	IFET
mollisols	xerolls	argixerolls	ultic argixerolls	I	F	E	U	IFEU
mollisols	xerolls	argixerolls	typic argixerolls	I	F	E	V	IFEV
mollisols	xerolls	haploxerolls	aridic lithic haploxerolls	I	F	F	A	IFFA
mollisols	xerolls	haploxerolls	lithic ultic haploxerolls	I	F	F	B	IFFB
mollisols	xerolls	haploxerolls	lithic haploxerolls	I	F	F	C	IFFC
mollisols	xerolls	haploxerolls	torrertic haploxerolls	I	F	F	D	IFFD
mollisols	xerolls	haploxerolls	vertic haploxerolls	I	F	F	E	IFFE
mollisols	xerolls	haploxerolls	andic haploxerolls	I	F	F	F	IFFF
mollisols	xerolls	haploxerolls	vitritorrandic haploxerolls	I	F	F	G	IFFG
mollisols	xerolls	haploxerolls	vitrandic haploxerolls	I	F	F	H	IFFH
mollisols	xerolls	haploxerolls	aquic cumulic haploxerolls	I	F	F	I	IFFI
mollisols	xerolls	haploxerolls	cumulic ultic haploxerolls	I	F	F	J	IFFJ
mollisols	xerolls	haploxerolls	cumulic haploxerolls	I	F	F	K	IFFK
mollisols	xerolls	haploxerolls	fluvaquentic haploxerolls	I	F	F	L	IFFL
mollisols	xerolls	haploxerolls	aquic duric haploxerolls	I	F	F	M	IFFM
mollisols	xerolls	haploxerolls	aquultic haploxerolls	I	F	F	N	IFFN
mollisols	xerolls	haploxerolls	aquic haploxerolls	I	F	F	O	IFFO
mollisols	xerolls	haploxerolls	oxyaquic haploxerolls	I	F	F	P	IFFP
mollisols	xerolls	haploxerolls	calcic pachic haploxerolls	I	F	F	Q	IFFQ
mollisols	xerolls	haploxerolls	pachic ultic haploxerolls	I	F	F	R	IFFR
mollisols	xerolls	haploxerolls	pachic haploxerolls	I	F	F	S	IFFS
mollisols	xerolls	haploxerolls	torrifluventic haploxerolls	I	F	F	T	IFFT
mollisols	xerolls	haploxerolls	duridic haploxerolls	I	F	F	U	IFFU
mollisols	xerolls	haploxerolls	calcidic haploxerolls	I	F	F	V	IFFV
mollisols	xerolls	haploxerolls	torripsammentic haploxerolls	I	F	F	W	IFFW
mollisols	xerolls	haploxerolls	torriorthentic haploxerolls	I	F	F	X	IFFX
mollisols	xerolls	haploxerolls	aridic haploxerolls	I	F	F	Y	IFFY
mollisols	xerolls	haploxerolls	duric haploxerolls	I	F	F	Z	IFFZ
mollisols	xerolls	haploxerolls	psammentic haploxerolls	I	F	F	Z	IFFZ
mollisols	xerolls	haploxerolls	fluventic haploxerolls	I	F	F	Z	IFFZ
mollisols	xerolls	haploxerolls	vermic haploxerolls	I	F	F	Z	IFFZ
mollisols	xerolls	haploxerolls	calcic haploxerolls	I	F	F	Z	IFFZ
mollisols	xerolls	haploxerolls	entic ultic haploxerolls	I	F	F	Z	IFFZ
mollisols	xerolls	haploxerolls	ultic haploxerolls	I	F	F	Z	IFFZ
mollisols	xerolls	haploxerolls	entic haploxerolls	I	F	F	Z	IFFZ
mollisols	xerolls	haploxerolls	typic haploxerolls	I	F	F	Z	IFFZ
mollisols	ustolls	durustolls	natric durustolls	I	G	A	A	IGAA
mollisols	ustolls	durustolls	haploduridic durustolls	I	G	A	B	IGAB
mollisols	ustolls	durustolls	argiduridic durustolls	I	G	A	C	IGAC
mollisols	ustolls	durustolls	entic durustolls	I	G	A	D	IGAD
mollisols	ustolls	durustolls	haplic durustolls	I	G	A	E	IGAE
mollisols	ustolls	durustolls	typic durustolls	I	G	A	F	IGAF
mollisols	ustolls	natrustolls	leptic torrertic natrustolls	I	G	B	A	IGBA
mollisols	ustolls	natrustolls	torrertic natrustolls	I	G	B	B	IGBB
mollisols	ustolls	natrustolls	leptic vertic natrustolls	I	G	B	C	IGBC
mollisols	ustolls	natrustolls	glossic vertic natrustolls	I	G	B	D	IGBD
mollisols	ustolls	natrustolls	vertic natrustolls	I	G	B	E	IGBE
mollisols	ustolls	natrustolls	aridic leptic natrustolls	I	G	B	F	IGBF
mollisols	ustolls	natrustolls	leptic natrustolls	I	G	B	G	IGBG
mollisols	ustolls	natrustolls	aquic natrustolls	I	G	B	H	IGBH
mollisols	ustolls	natrustolls	aridic natrustolls	I	G	B	I	IGBI
mollisols	ustolls	natrustolls	duric natrustolls	I	G	B	J	IGBJ
mollisols	ustolls	natrustolls	glossic natrustolls	I	G	B	K	IGBK
mollisols	ustolls	natrustolls	typic natrustolls	I	G	B	L	IGBL
mollisols	ustolls	calciustolls	salidic calciustolls	I	G	C	A	IGCA

mollisols	ustolls	calciustolls	lithic petrocalcic calciustolls	I	G	C	B	IGCB
mollisols	ustolls	calciustolls	lithic calciustolls	I	G	C	C	IGCC
mollisols	ustolls	calciustolls	torrertic calciustolls	I	G	C	D	IGCD
mollisols	ustolls	calciustolls	udertic calciustolls	I	G	C	E	IGCE
mollisols	ustolls	calciustolls	vertic calciustolls	I	G	C	F	IGCF
mollisols	ustolls	calciustolls	petrocalcic calciustolls	I	G	C	G	IGCG
mollisols	ustolls	calciustolls	gypsic calciustolls	I	G	C	H	IGCH
mollisols	ustolls	calciustolls	aquic calciustolls	I	G	C	I	IGCI
mollisols	ustolls	calciustolls	oxyaquic calciustolls	I	G	C	J	IGCJ
mollisols	ustolls	calciustolls	pachic calciustolls	I	G	C	K	IGCK
mollisols	ustolls	calciustolls	aridic calciustolls	I	G	C	L	IGCL
mollisols	ustolls	calciustolls	udic calciustolls	I	G	C	M	IGCM
mollisols	ustolls	calciustolls	typic calciustolls	I	G	C	N	IGCN
mollisols	ustolls	paleustolls	torrertic paleustolls	I	G	D	A	IGDA
mollisols	ustolls	paleustolls	udertic paleustolls	I	G	D	B	IGDB
mollisols	ustolls	paleustolls	vertic paleustolls	I	G	D	C	IGDC
mollisols	ustolls	paleustolls	aquic paleustolls	I	G	D	D	IGDD
mollisols	ustolls	paleustolls	pachic paleustolls	I	G	D	E	IGDE
mollisols	ustolls	paleustolls	petrocalcic paleustolls	I	G	D	F	IGDF
mollisols	ustolls	paleustolls	calcidic paleustolls	I	G	D	G	IGDG
mollisols	ustolls	paleustolls	aridic paleustolls	I	G	D	H	IGDH
mollisols	ustolls	paleustolls	udic paleustolls	I	G	D	I	IGDI
mollisols	ustolls	paleustolls	calcic paleustolls	I	G	D	J	IGDJ
mollisols	ustolls	paleustolls	entic paleustolls	I	G	D	K	IGDK
mollisols	ustolls	paleustolls	typic paleustolls	I	G	D	L	IGDL
mollisols	ustolls	argiustolls	aridic lithic argiustolls	I	G	E	A	IGEA
mollisols	ustolls	argiustolls	alfic lithic argiustolls	I	G	E	B	IGEB
mollisols	ustolls	argiustolls	lithic argiustolls	I	G	E	C	IGEC
mollisols	ustolls	argiustolls	aquertic argiustolls	I	G	E	D	IGED
mollisols	ustolls	argiustolls	torrertic argiustolls	I	G	E	E	IGEE
mollisols	ustolls	argiustolls	pachic udertic argiustolls	I	G	E	F	IGEF
mollisols	ustolls	argiustolls	udertic argiustolls	I	G	E	G	IGEG
mollisols	ustolls	argiustolls	pachic vertic argiustolls	I	G	E	H	IGEH
mollisols	ustolls	argiustolls	vertic argiustolls	I	G	E	I	IGEI
mollisols	ustolls	argiustolls	andic argiustolls	I	G	E	J	IGEJ
mollisols	ustolls	argiustolls	vitritorrandic argiustolls	I	G	E	K	IGEK
mollisols	ustolls	argiustolls	vitrandic argiustolls	I	G	E	L	IGEL
mollisols	ustolls	argiustolls	aquic argiustolls	I	G	E	M	IGEM
mollisols	ustolls	argiustolls	oxyaquic argiustolls	I	G	E	N	IGEN
mollisols	ustolls	argiustolls	pachic argiustolls	I	G	E	O	IGEO
mollisols	ustolls	argiustolls	alfic argiustolls	I	G	E	P	IGEP
mollisols	ustolls	argiustolls	calcidic argiustolls	I	G	E	Q	IGEQ
mollisols	ustolls	argiustolls	aridic argiustolls	I	G	E	R	IGER
mollisols	ustolls	argiustolls	udic argiustolls	I	G	E	S	IGES
mollisols	ustolls	argiustolls	duric argiustolls	I	G	E	T	IGET
mollisols	ustolls	argiustolls	typic argiustolls	I	G	E	U	IGEU
mollisols	ustolls	vermustolls	lithic vermustolls	I	G	F	A	IGFA
mollisols	ustolls	vermustolls	aquic vermustolls	I	G	F	B	IGFB
mollisols	ustolls	vermustolls	pachic vermustolls	I	G	F	C	IGFC
mollisols	ustolls	vermustolls	entic vermustolls	I	G	F	D	IGFD
mollisols	ustolls	vermustolls	typic vermustolls	I	G	F	E	IGFE
mollisols	ustolls	haplustolls	salidic haplustolls	I	G	G	A	IGGA
mollisols	ustolls	haplustolls	ruptic-lithic haplustolls	I	G	G	B	IGGB
mollisols	ustolls	haplustolls	aridic lithic haplustolls	I	G	G	C	IGGC
mollisols	ustolls	haplustolls	lithic haplustolls	I	G	G	D	IGGD
mollisols	ustolls	haplustolls	aquertic haplustolls	I	G	G	E	IGGE
mollisols	ustolls	haplustolls	torrertic haplustolls	I	G	G	F	IGGF
mollisols	ustolls	haplustolls	pachic udertic haplustolls	I	G	G	G	IGGG
mollisols	ustolls	haplustolls	udertic haplustolls	I	G	G	H	IGGH
mollisols	ustolls	haplustolls	pachic vertic haplustolls	I	G	G	I	IGGI

mollisols	ustolls	haplustolls	vertic haplustolls	I	G	G	J	IGGJ
mollisols	ustolls	haplustolls	torroxic haplustolls	I	G	G	K	IGGK
mollisols	ustolls	haplustolls	oxic haplustolls	I	G	G	L	IGGL
mollisols	ustolls	haplustolls	andic haplustolls	I	G	G	M	IGGM
mollisols	ustolls	haplustolls	vitritorrandic haplustolls	I	G	G	N	IGGN
mollisols	ustolls	haplustolls	vitrandic haplustolls	I	G	G	O	IGGO
mollisols	ustolls	haplustolls	aquic cumulic haplustolls	I	G	G	P	IGGP
mollisols	ustolls	haplustolls	cumulic haplustolls	I	G	G	Q	IGGQ
mollisols	ustolls	haplustolls	anthraquic haplustolls	I	G	G	R	IGGR
mollisols	ustolls	haplustolls	fluvaquentic haplustolls	I	G	G	S	IGGS
mollisols	ustolls	haplustolls	aquic haplustolls	I	G	G	T	IGGT
mollisols	ustolls	haplustolls	pachic haplustolls	I	G	G	U	IGGU
mollisols	ustolls	haplustolls	oxyaquic haplustolls	I	G	G	V	IGGV
mollisols	ustolls	haplustolls	torrifluventic haplustolls	I	G	G	W	IGGW
mollisols	ustolls	haplustolls	torriorthentic haplustolls	I	G	G	X	IGGX
mollisols	ustolls	haplustolls	aridic haplustolls	I	G	G	Y	IGGY
mollisols	ustolls	haplustolls	fluventic haplustolls	I	G	G	Z	IGGZ
mollisols	ustolls	haplustolls	duric haplustolls	I	G	G	Z	IGGZ
mollisols	ustolls	haplustolls	udorthentic haplustolls	I	G	G	Z	IGGZ
mollisols	ustolls	haplustolls	udic haplustolls	I	G	G	Z	IGGZ
mollisols	ustolls	haplustolls	entic haplustolls	I	G	G	Z	IGGZ
mollisols	ustolls	haplustolls	typic haplustolls	I	G	G	Z	IGGZ
mollisols	udolls	natrudolls	petrocalcic natrudolls	I	H	A	A	IHAA
mollisols	udolls	natrudolls	leptic vertic natrudolls	I	H	A	B	IHAB
mollisols	udolls	natrudolls	glossic vertic natrudolls	I	H	A	C	IHAC
mollisols	udolls	natrudolls	vertic natrudolls	I	H	A	D	IHAD
mollisols	udolls	natrudolls	leptic natrudolls	I	H	A	E	IHAE
mollisols	udolls	natrudolls	abruptic natrudolls	I	H	A	F	IHAF
mollisols	udolls	natrudolls	glossic natrudolls	I	H	A	G	IHAG
mollisols	udolls	natrudolls	calcic natrudolls	I	H	A	H	IHAH
mollisols	udolls	natrudolls	typic natrudolls	I	H	A	I	IHAI
mollisols	udolls	calciudolls	lithic calciudolls	I	H	B	A	IHBA
mollisols	udolls	calciudolls	vertic calciudolls	I	H	B	B	IHBB
mollisols	udolls	calciudolls	anthropic petrocalcic calciudolls	I	H	B	C	IHBC
mollisols	udolls	calciudolls	aquic calciudolls	I	H	B	D	IHBD
mollisols	udolls	calciudolls	fluventic calciudolls	I	H	B	E	IHBE
mollisols	udolls	calciudolls	typic calciudolls	I	H	B	F	IHBF
mollisols	udolls	paleudolls	vertic paleudolls	I	H	C	A	IHCA
mollisols	udolls	paleudolls	aquic pachic paleudolls	I	H	C	B	IHCB
mollisols	udolls	paleudolls	pachic paleudolls	I	H	C	C	IHCC
mollisols	udolls	paleudolls	aquic paleudolls	I	H	C	D	IHCD
mollisols	udolls	paleudolls	oxyaquic paleudolls	I	H	C	E	IHCE
mollisols	udolls	paleudolls	calcic paleudolls	I	H	C	F	IHCF
mollisols	udolls	paleudolls	typic paleudolls	I	H	C	G	IHCG
mollisols	udolls	argiudolls	lithic argiudolls	I	H	D	A	IHDA
mollisols	udolls	argiudolls	petrocalcic argiudolls	I	H	D	B	IHDB
mollisols	udolls	argiudolls	aquertic argiudolls	I	H	D	C	IHDC
mollisols	udolls	argiudolls	oxyaquic vertic argiudolls	I	H	D	D	IHDD
mollisols	udolls	argiudolls	pachic vertic argiudolls	I	H	D	E	IHDE
mollisols	udolls	argiudolls	alfic vertic argiudolls	I	H	D	F	IHDF
mollisols	udolls	argiudolls	vertic argiudolls	I	H	D	G	IHDG
mollisols	udolls	argiudolls	andic argiudolls	I	H	D	H	IHDH
mollisols	udolls	argiudolls	vitrandic argiudolls	I	H	D	I	IHDI
mollisols	udolls	argiudolls	aquic pachic argiudolls	I	H	D	J	IHDJ
mollisols	udolls	argiudolls	pachic argiudolls	I	H	D	K	IHDK
mollisols	udolls	argiudolls	aquic argiudolls	I	H	D	L	IHDL
mollisols	udolls	argiudolls	oxyaquic argiudolls	I	H	D	M	IHDM
mollisols	udolls	argiudolls	lamellic argiudolls	I	H	D	N	IHDN
mollisols	udolls	argiudolls	psammentic argiudolls	I	H	D	O	IHDO
mollisols	udolls	argiudolls	arenic argiudolls	I	H	D	P	IHDP

mollisols	udolls	argiudolls	abruptic argiudolls	I	H	D	Q	IHDQ
mollisols	udolls	argiudolls	alfic argiudolls	I	H	D	R	IHDR
mollisols	udolls	argiudolls	oxic argiudolls	I	H	D	S	IHDS
mollisols	udolls	argiudolls	calcic argiudolls	I	H	D	T	IHDT
mollisols	udolls	argiudolls	typic argiudolls	I	H	D	U	IHDU
mollisols	udolls	vermudolls	lithic vermudolls	I	H	E	A	IHEA
mollisols	udolls	vermudolls	haplic vermudolls	I	H	E	B	IHEB
mollisols	udolls	vermudolls	typic vermudolls	I	H	E	C	IHEC
mollisols	udolls	hapludolls	lithic hapludolls	I	H	F	A	IHFA
mollisols	udolls	hapludolls	petrocalcic hapludolls	I	H	F	B	IHFB
mollisols	udolls	hapludolls	aquertic hapludolls	I	H	F	C	IHFC
mollisols	udolls	hapludolls	pachic vertic hapludolls	I	H	F	D	IHFD
mollisols	udolls	hapludolls	vertic hapludolls	I	H	F	E	IHFE
mollisols	udolls	hapludolls	andic hapludolls	I	H	F	F	IHFF
mollisols	udolls	hapludolls	vitrandic hapludolls	I	H	F	G	IHFG
mollisols	udolls	hapludolls	aquic cumulic hapludolls	I	H	F	H	IHFH
mollisols	udolls	hapludolls	cumulic hapludolls	I	H	F	I	IHFI
mollisols	udolls	hapludolls	fluvaquentic hapludolls	I	H	F	J	IHFJ
mollisols	udolls	hapludolls	fluventic hapludolls	I	H	F	K	IHFK
mollisols	udolls	hapludolls	aquic pachic hapludolls	I	H	F	L	IHFL
mollisols	udolls	hapludolls	pachic hapludolls	I	H	F	M	IHFM
mollisols	udolls	hapludolls	aquic hapludolls	I	H	F	N	IHFN
mollisols	udolls	hapludolls	oxyaquic hapludolls	I	H	F	O	IHFO
mollisols	udolls	hapludolls	vermic hapludolls	I	H	F	P	IHFP
mollisols	udolls	hapludolls	calcic hapludolls	I	H	F	Q	IHFQ
mollisols	udolls	hapludolls	entic hapludolls	I	H	F	R	IHFR
mollisols	udolls	hapludolls	typic hapludolls	I	H	F	S	IHFS
alfisols	aqualfs	cryaqualfs	typic cryaqualfs	J	A	A	A	JAAA
alfisols	aqualfs	plinthaqualfs	typic plinthaqualfs	J	A	B	A	JABA
alfisols	aqualfs	duraqualfs	typic duraqualfs	J	A	C	A	JACA
alfisols	aqualfs	natraqualfs	vertic natraqualfs	J	A	D	A	JADA
alfisols	aqualfs	natraqualfs	vermic natraqualfs	J	A	D	B	JADB
alfisols	aqualfs	natraqualfs	albic glossic natraqualfs	J	A	D	C	JADC
alfisols	aqualfs	natraqualfs	albic natraqualfs	J	A	D	D	JADD
alfisols	aqualfs	natraqualfs	glossic natraqualfs	J	A	D	E	JADE
alfisols	aqualfs	natraqualfs	mollic natraqualfs	J	A	D	F	JADF
alfisols	aqualfs	natraqualfs	typic natraqualfs	J	A	D	G	JADG
alfisols	aqualfs	fragiaqualfs	vermic fragiaqualfs	J	A	E	A	JAEA
alfisols	aqualfs	fragiaqualfs	aeric fragiaqualfs	J	A	E	B	JAEB
alfisols	aqualfs	fragiaqualfs	plinthic fragiaqualfs	J	A	E	C	JAEC
alfisols	aqualfs	fragiaqualfs	humic fragiaqualfs	J	A	E	D	JAED
alfisols	aqualfs	fragiaqualfs	typic fragiaqualfs	J	A	E	E	JAEE
alfisols	aqualfs	kandiaqualfs	arenic kandiaqualfs	J	A	F	A	JAFA
alfisols	aqualfs	kandiaqualfs	grossarenic kandiaqualfs	J	A	F	B	JAFB
alfisols	aqualfs	kandiaqualfs	plinthic kandiaqualfs	J	A	F	C	JAFC
alfisols	aqualfs	kandiaqualfs	aeric umbric kandiaqualfs	J	A	F	D	JAFD
alfisols	aqualfs	kandiaqualfs	aeric kandiaqualfs	J	A	F	E	JAFE
alfisols	aqualfs	kandiaqualfs	umbric kandiaqualfs	J	A	F	F	JAFF
alfisols	aqualfs	kandiaqualfs	typic kandiaqualfs	J	A	F	G	JAFG
alfisols	aqualfs	vermaqualfs	natric vermaqualfs	J	A	G	A	JAGA
alfisols	aqualfs	vermaqualfs	typic vermaqualfs	J	A	G	B	JAGB
alfisols	aqualfs	albaqualfs	arenic albaqualfs	J	A	H	A	JAHA
alfisols	aqualfs	albaqualfs	aeric vertic albaqualfs	J	A	H	B	JAHB
alfisols	aqualfs	albaqualfs	chromic vertic albaqualfs	J	A	H	C	JAHC
alfisols	aqualfs	albaqualfs	vertic albaqualfs	J	A	H	D	JAHD
alfisols	aqualfs	albaqualfs	udollic albaqualfs	J	A	H	E	JAHE
alfisols	aqualfs	albaqualfs	aeric albaqualfs	J	A	H	F	JAHF
alfisols	aqualfs	albaqualfs	aquandic albaqualfs	J	A	H	G	JAHG
alfisols	aqualfs	albaqualfs	mollic albaqualfs	J	A	H	H	JAHH
alfisols	aqualfs	albaqualfs	umbric albaqualfs	J	A	H	I	JAHI

alfisols	aqualfs	albaqualfs	typic albaqualfs	J	A	H	J	JAHJ
alfisols	aqualfs	glossaqualfs	histic glossaqualfs	J	A	I	A	JAIA
alfisols	aqualfs	glossaqualfs	arenic glossaqualfs	J	A	I	B	JAIB
alfisols	aqualfs	glossaqualfs	aeric fragic glossaqualfs	J	A	I	C	JAIC
alfisols	aqualfs	glossaqualfs	fragic glossaqualfs	J	A	I	D	JAID
alfisols	aqualfs	glossaqualfs	aeric glossaqualfs	J	A	I	E	JAIE
alfisols	aqualfs	glossaqualfs	mollic glossaqualfs	J	A	I	F	JAIF
alfisols	aqualfs	glossaqualfs	typic glossaqualfs	J	A	I	G	JAIG
alfisols	aqualfs	epiaqualfs	aeric chromic vertic epiaqualfs	J	A	J	A	JAJA
alfisols	aqualfs	epiaqualfs	aeric vertic epiaqualfs	J	A	J	B	JAJB
alfisols	aqualfs	epiaqualfs	chromic vertic epiaqualfs	J	A	J	C	JAJC
alfisols	aqualfs	epiaqualfs	vertic epiaqualfs	J	A	J	D	JAJD
alfisols	aqualfs	epiaqualfs	aquandic epiaqualfs	J	A	J	E	JAJE
alfisols	aqualfs	epiaqualfs	aeric fragic epiaqualfs	J	A	J	F	JAJF
alfisols	aqualfs	epiaqualfs	fragic epiaqualfs	J	A	J	G	JAJG
alfisols	aqualfs	epiaqualfs	arenic epiaqualfs	J	A	J	H	JAJH
alfisols	aqualfs	epiaqualfs	grossarenic epiaqualfs	J	A	J	I	JAJI
alfisols	aqualfs	epiaqualfs	aeric umbric epiaqualfs	J	A	J	J	JAJJ
alfisols	aqualfs	epiaqualfs	udollic epiaqualfs	J	A	J	K	JAJK
alfisols	aqualfs	epiaqualfs	aeric epiaqualfs	J	A	J	L	JAJL
alfisols	aqualfs	epiaqualfs	mollic epiaqualfs	J	A	J	M	JAJM
alfisols	aqualfs	epiaqualfs	umbric epiaqualfs	J	A	J	N	JAJN
alfisols	aqualfs	epiaqualfs	typic epiaqualfs	J	A	J	O	JAJO
alfisols	aqualfs	endoaqualfs	aquandic endoaqualfs	J	A	K	A	JAKA
alfisols	aqualfs	endoaqualfs	chromic vertic endoaqualfs	J	A	K	B	JAKB
alfisols	aqualfs	endoaqualfs	vertic endoaqualfs	J	A	K	C	JAKC
alfisols	aqualfs	endoaqualfs	aeric fragic endoaqualfs	J	A	K	D	JAKD
alfisols	aqualfs	endoaqualfs	fragic endoaqualfs	J	A	K	E	JAKE
alfisols	aqualfs	endoaqualfs	arenic endoaqualfs	J	A	K	F	JAKF
alfisols	aqualfs	endoaqualfs	grossarenic endoaqualfs	J	A	K	G	JAKG
alfisols	aqualfs	endoaqualfs	udollic endoaqualfs	J	A	K	H	JAKH
alfisols	aqualfs	endoaqualfs	aeric umbric endoaqualfs	J	A	K	I	JAKI
alfisols	aqualfs	endoaqualfs	aeric endoaqualfs	J	A	K	J	JAKJ
alfisols	aqualfs	endoaqualfs	mollic endoaqualfs	J	A	K	K	JAKK
alfisols	aqualfs	endoaqualfs	umbric endoaqualfs	J	A	K	L	JAKL
alfisols	aqualfs	endoaqualfs	typic endoaqualfs	J	A	K	M	JAKM
alfisols	cryalfs	palecryalfs	andic palecryalfs	J	B	A	A	JBAA
alfisols	cryalfs	palecryalfs	vitrandic palecryalfs	J	B	A	B	JBAB
alfisols	cryalfs	palecryalfs	aquic palecryalfs	J	B	A	C	JBAC
alfisols	cryalfs	palecryalfs	oxyaquic palecryalfs	J	B	A	D	JBAD
alfisols	cryalfs	palecryalfs	xeric palecryalfs	J	B	A	E	JBAE
alfisols	cryalfs	palecryalfs	ustic palecryalfs	J	B	A	F	JBAF
alfisols	cryalfs	palecryalfs	mollic palecryalfs	J	B	A	G	JBAG
alfisols	cryalfs	palecryalfs	umbric palecryalfs	J	B	A	H	JBAH
alfisols	cryalfs	palecryalfs	typic palecryalfs	J	B	A	I	JBAI
alfisols	cryalfs	glossocryalfs	lithic glossocryalfs	J	B	B	A	JBBA
alfisols	cryalfs	glossocryalfs	vertic glossocryalfs	J	B	B	B	JBBB
alfisols	cryalfs	glossocryalfs	andic glossocryalfs	J	B	B	C	JBBC
alfisols	cryalfs	glossocryalfs	vitrandic glossocryalfs	J	B	B	D	JBBD
alfisols	cryalfs	glossocryalfs	aquic glossocryalfs	J	B	B	E	JBBE
alfisols	cryalfs	glossocryalfs	oxyaquic glossocryalfs	J	B	B	F	JBBF
alfisols	cryalfs	glossocryalfs	fragic glossocryalfs	J	B	B	G	JBBG
alfisols	cryalfs	glossocryalfs	xerollic glossocryalfs	J	B	B	H	JBBH
alfisols	cryalfs	glossocryalfs	umbric xeric glossocryalfs	J	B	B	I	JBBI
alfisols	cryalfs	glossocryalfs	ustollic glossocryalfs	J	B	B	J	JBBJ
alfisols	cryalfs	glossocryalfs	xeric glossocryalfs	J	B	B	K	JBBK
alfisols	cryalfs	glossocryalfs	ustic glossocryalfs	J	B	B	L	JBBL
alfisols	cryalfs	glossocryalfs	mollic glossocryalfs	J	B	B	M	JBBM
alfisols	cryalfs	glossocryalfs	umbric glossocryalfs	J	B	B	N	JBBN
alfisols	cryalfs	glossocryalfs	eutric glossocryalfs	J	B	B	O	JBBO

alfisols	cryalfs	glossocryalfs	typic glossocryalfs	J	B	B	P	JBBP
alfisols	cryalfs	haplocryalfs	lithic haplocryalfs	J	B	C	A	JBCA
alfisols	cryalfs	haplocryalfs	vertic haplocryalfs	J	B	C	B	JBCB
alfisols	cryalfs	haplocryalfs	andic haplocryalfs	J	B	C	C	JBCC
alfisols	cryalfs	haplocryalfs	vitrandic haplocryalfs	J	B	C	D	JBCD
alfisols	cryalfs	haplocryalfs	aquic haplocryalfs	J	B	C	E	JBCE
alfisols	cryalfs	haplocryalfs	oxyaquic haplocryalfs	J	B	C	F	JBCF
alfisols	cryalfs	haplocryalfs	lamellic haplocryalfs	J	B	C	G	JBCG
alfisols	cryalfs	haplocryalfs	psammentic haplocryalfs	J	B	C	H	JBCH
alfisols	cryalfs	haplocryalfs	inceptic haplocryalfs	J	B	C	I	JBCI
alfisols	cryalfs	haplocryalfs	xerollic haplocryalfs	J	B	C	J	JBCJ
alfisols	cryalfs	haplocryalfs	umbric xeric haplocryalfs	J	B	C	K	JBCK
alfisols	cryalfs	haplocryalfs	ustollic haplocryalfs	J	B	C	L	JBCL
alfisols	cryalfs	haplocryalfs	xeric haplocryalfs	J	B	C	M	JBCM
alfisols	cryalfs	haplocryalfs	ustic haplocryalfs	J	B	C	N	JBCN
alfisols	cryalfs	haplocryalfs	mollic haplocryalfs	J	B	C	O	JBCO
alfisols	cryalfs	haplocryalfs	umbric haplocryalfs	J	B	C	P	JBCP
alfisols	cryalfs	haplocryalfs	eutric haplocryalfs	J	B	C	Q	JBCQ
alfisols	cryalfs	haplocryalfs	typic haplocryalfs	J	B	C	R	JBCR
alfisols	ustalfs	durustalfs	typic durustalfs	J	C	A	A	JCAA
alfisols	ustalfs	plinthustalfs	typic plinthustalfs	J	C	B	A	JCBA
alfisols	ustalfs	natrustalfs	salidic natrustalfs	J	C	C	A	JCCA
alfisols	ustalfs	natrustalfs	leptic torrertic natrustalfs	J	C	C	B	JCCB
alfisols	ustalfs	natrustalfs	torrertic natrustalfs	J	C	C	C	JCCC
alfisols	ustalfs	natrustalfs	aquertic natrustalfs	J	C	C	D	JCCD
alfisols	ustalfs	natrustalfs	aridic leptic natrustalfs	J	C	C	E	JCCE
alfisols	ustalfs	natrustalfs	vertic natrustalfs	J	C	C	F	JCCF
alfisols	ustalfs	natrustalfs	aquic arenic natrustalfs	J	C	C	G	JCCG
alfisols	ustalfs	natrustalfs	aquic natrustalfs	J	C	C	H	JCCH
alfisols	ustalfs	natrustalfs	arenic natrustalfs	J	C	C	I	JCCI
alfisols	ustalfs	natrustalfs	petrocalcic natrustalfs	J	C	C	J	JCCJ
alfisols	ustalfs	natrustalfs	leptic natrustalfs	J	C	C	K	JCCK
alfisols	ustalfs	natrustalfs	haplargidic natrustalfs	J	C	C	L	JCCL
alfisols	ustalfs	natrustalfs	aridic glossic natrustalfs	J	C	C	M	JCCM
alfisols	ustalfs	natrustalfs	aridic natrustalfs	J	C	C	N	JCCN
alfisols	ustalfs	natrustalfs	mollic natrustalfs	J	C	C	O	JCCO
alfisols	ustalfs	natrustalfs	typic natrustalfs	J	C	C	P	JCCP
alfisols	ustalfs	kandiustalfs	grossarenic kandiustalfs	J	C	D	A	JCDA
alfisols	ustalfs	kandiustalfs	aquic arenic kandiustalfs	J	C	D	B	JCDB
alfisols	ustalfs	kandiustalfs	plinthic kandiustalfs	J	C	D	C	JCDC
alfisols	ustalfs	kandiustalfs	aquic kandiustalfs	J	C	D	D	JCDD
alfisols	ustalfs	kandiustalfs	arenic aridic kandiustalfs	J	C	D	E	JCDE
alfisols	ustalfs	kandiustalfs	arenic kandiustalfs	J	C	D	F	JCDF
alfisols	ustalfs	kandiustalfs	aridic kandiustalfs	J	C	D	G	JCDG
alfisols	ustalfs	kandiustalfs	udic kandiustalfs	J	C	D	H	JCDH
alfisols	ustalfs	kandiustalfs	rhodic kandiustalfs	J	C	D	I	JCDI
alfisols	ustalfs	kandiustalfs	typic kandiustalfs	J	C	D	J	JCDJ
alfisols	ustalfs	kanhaplustalfs	lithic kanhaplustalfs	J	C	E	A	JCEA
alfisols	ustalfs	kanhaplustalfs	aquic kanhaplustalfs	J	C	E	B	JCEB
alfisols	ustalfs	kanhaplustalfs	aridic kanhaplustalfs	J	C	E	C	JCEC
alfisols	ustalfs	kanhaplustalfs	udic kanhaplustalfs	J	C	E	D	JCED
alfisols	ustalfs	kanhaplustalfs	rhodic kanhaplustalfs	J	C	E	E	JCEE
alfisols	ustalfs	kanhaplustalfs	typic kanhaplustalfs	J	C	E	F	JCEF
alfisols	ustalfs	paleustalfs	aquertic paleustalfs	J	C	F	A	JCFA
alfisols	ustalfs	paleustalfs	oxyaquic vertic paleustalfs	J	C	F	B	JCFB
alfisols	ustalfs	paleustalfs	udertic paleustalfs	J	C	F	C	JCFC
alfisols	ustalfs	paleustalfs	vertic paleustalfs	J	C	F	D	JCFD
alfisols	ustalfs	paleustalfs	aquic arenic paleustalfs	J	C	F	E	JCFE
alfisols	ustalfs	paleustalfs	aquic paleustalfs	J	C	F	F	JCFF
alfisols	ustalfs	paleustalfs	oxyaquic paleustalfs	J	C	F	G	JCFG

alfisols	ustalfs	paleustalfs	lamellic paleustalfs	J	C	F	H	JCFH
alfisols	ustalfs	paleustalfs	psammentic paleustalfs	J	C	F	I	JCFI
alfisols	ustalfs	paleustalfs	arenic aridic paleustalfs	J	C	F	J	JCFJ
alfisols	ustalfs	paleustalfs	grossarenic paleustalfs	J	C	F	K	JCFK
alfisols	ustalfs	paleustalfs	arenic paleustalfs	J	C	F	L	JCFL
alfisols	ustalfs	paleustalfs	plinthic paleustalfs	J	C	F	M	JCFM
alfisols	ustalfs	paleustalfs	petrocalcic paleustalfs	J	C	F	N	JCFN
alfisols	ustalfs	paleustalfs	calcidic paleustalfs	J	C	F	O	JCFO
alfisols	ustalfs	paleustalfs	aridic paleustalfs	J	C	F	P	JCFP
alfisols	ustalfs	paleustalfs	kandic paleustalfs	J	C	F	Q	JCFQ
alfisols	ustalfs	paleustalfs	rhodic paleustalfs	J	C	F	R	JCFR
alfisols	ustalfs	paleustalfs	ultic paleustalfs	J	C	F	S	JCFS
alfisols	ustalfs	paleustalfs	udic paleustalfs	J	C	F	T	JCFT
alfisols	ustalfs	paleustalfs	typic paleustalfs	J	C	F	U	JCFU
alfisols	ustalfs	rhodustalfs	lithic rhodustalfs	J	C	G	A	JCGA
alfisols	ustalfs	rhodustalfs	kanhaplic rhodustalfs	J	C	G	B	JCGB
alfisols	ustalfs	rhodustalfs	udic rhodustalfs	J	C	G	C	JCGC
alfisols	ustalfs	rhodustalfs	typic rhodustalfs	J	C	G	D	JCGD
alfisols	ustalfs	haplustalfs	lithic haplustalfs	J	C	H	A	JCHA
alfisols	ustalfs	haplustalfs	aquertic haplustalfs	J	C	H	B	JCHB
alfisols	ustalfs	haplustalfs	oxyaquic vertic haplustalfs	J	C	H	C	JCHC
alfisols	ustalfs	haplustalfs	torrertic haplustalfs	J	C	H	D	JCHD
alfisols	ustalfs	haplustalfs	udertic haplustalfs	J	C	H	E	JCHE
alfisols	ustalfs	haplustalfs	vertic haplustalfs	J	C	H	F	JCHF
alfisols	ustalfs	haplustalfs	aquic arenic haplustalfs	J	C	H	G	JCHG
alfisols	ustalfs	haplustalfs	aquultic haplustalfs	J	C	H	H	JCHH
alfisols	ustalfs	haplustalfs	aquic haplustalfs	J	C	H	I	JCHI
alfisols	ustalfs	haplustalfs	oxyaquic haplustalfs	J	C	H	J	JCHJ
alfisols	ustalfs	haplustalfs	vitrandic haplustalfs	J	C	H	K	JCHK
alfisols	ustalfs	haplustalfs	lamellic haplustalfs	J	C	H	L	JCHL
alfisols	ustalfs	haplustalfs	psammentic haplustalfs	J	C	H	M	JCHM
alfisols	ustalfs	haplustalfs	arenic aridic haplustalfs	J	C	H	N	JCHN
alfisols	ustalfs	haplustalfs	arenic haplustalfs	J	C	H	O	JCHO
alfisols	ustalfs	haplustalfs	calcidic haplustalfs	J	C	H	P	JCHP
alfisols	ustalfs	haplustalfs	aridic haplustalfs	J	C	H	Q	JCHQ
alfisols	ustalfs	haplustalfs	kanhaplic haplustalfs	J	C	H	R	JCHR
alfisols	ustalfs	haplustalfs	inceptic haplustalfs	J	C	H	S	JCHS
alfisols	ustalfs	haplustalfs	calcic udic haplustalfs	J	C	H	T	JCHT
alfisols	ustalfs	haplustalfs	ultic haplustalfs	J	C	H	U	JCHU
alfisols	ustalfs	haplustalfs	calcic haplustalfs	J	C	H	V	JCHV
alfisols	ustalfs	haplustalfs	udic haplustalfs	J	C	H	W	JCHW
alfisols	ustalfs	haplustalfs	typic haplustalfs	J	C	H	X	JCHX
alfisols	xeralfs	durixeralfs	natric durixeralfs	J	D	A	A	JDAA
alfisols	xeralfs	durixeralfs	vertic durixeralfs	J	D	A	B	JDAB
alfisols	xeralfs	durixeralfs	aquic durixeralfs	J	D	A	C	JDAC
alfisols	xeralfs	durixeralfs	abruptic haplic durixeralfs	J	D	A	D	JDAD
alfisols	xeralfs	durixeralfs	abruptic durixeralfs	J	D	A	E	JDAE
alfisols	xeralfs	durixeralfs	haplic durixeralfs	J	D	A	F	JDAF
alfisols	xeralfs	durixeralfs	typic durixeralfs	J	D	A	G	JDAG
alfisols	xeralfs	natrixeralfs	vertic natrixeralfs	J	D	B	A	JDBA
alfisols	xeralfs	natrixeralfs	aquic natrixeralfs	J	D	B	B	JDBB
alfisols	xeralfs	natrixeralfs	typic natrixeralfs	J	D	B	C	JDBC
alfisols	xeralfs	fragixeralfs	andic fragixeralfs	J	D	C	A	JDCA
alfisols	xeralfs	fragixeralfs	vitrandic fragixeralfs	J	D	C	B	JDCB
alfisols	xeralfs	fragixeralfs	mollic fragixeralfs	J	D	C	C	JDCC
alfisols	xeralfs	fragixeralfs	aquic fragixeralfs	J	D	C	D	JDCD
alfisols	xeralfs	fragixeralfs	inceptic fragixeralfs	J	D	C	E	JDCE
alfisols	xeralfs	fragixeralfs	typic fragixeralfs	J	D	C	F	JDCF
alfisols	xeralfs	plinthoxeralfs	typic plinthoxeralfs	J	D	D	A	JDDA
alfisols	xeralfs	rhodoxeralfs	lithic rhodoxeralfs	J	D	E	A	JDEA

alfisols	xeralfs	rhodoxeralfs	vertic rhodoxeralfs	J	D	E	B	JDEB
alfisols	xeralfs	rhodoxeralfs	petrocalcic rhodoxeralfs	J	D	E	C	JDEC
alfisols	xeralfs	rhodoxeralfs	calcic rhodoxeralfs	J	D	E	D	JDED
alfisols	xeralfs	rhodoxeralfs	inceptic rhodoxeralfs	J	D	E	E	JDEE
alfisols	xeralfs	rhodoxeralfs	typic rhodoxeralfs	J	D	E	F	JDEF
alfisols	xeralfs	palexeralfs	vertic palexeralfs	J	D	F	A	JDFA
alfisols	xeralfs	palexeralfs	aquandic palexeralfs	J	D	F	B	JDFB
alfisols	xeralfs	palexeralfs	andic palexeralfs	J	D	F	C	JDFC
alfisols	xeralfs	palexeralfs	vitrandic palexeralfs	J	D	F	D	JDFD
alfisols	xeralfs	palexeralfs	fragiaquic palexeralfs	J	D	F	E	JDFE
alfisols	xeralfs	palexeralfs	aquic palexeralfs	J	D	F	F	JDFF
alfisols	xeralfs	palexeralfs	petrocalcic palexeralfs	J	D	F	G	JDFG
alfisols	xeralfs	palexeralfs	lamellic palexeralfs	J	D	F	H	JDFH
alfisols	xeralfs	palexeralfs	psammentic palexeralfs	J	D	F	I	JDFI
alfisols	xeralfs	palexeralfs	arenic palexeralfs	J	D	F	J	JDFJ
alfisols	xeralfs	palexeralfs	natric palexeralfs	J	D	F	K	JDFK
alfisols	xeralfs	palexeralfs	fragic palexeralfs	J	D	F	L	JDFL
alfisols	xeralfs	palexeralfs	calcic palexeralfs	J	D	F	M	JDFM
alfisols	xeralfs	palexeralfs	plinthic palexeralfs	J	D	F	N	JDFN
alfisols	xeralfs	palexeralfs	ultic palexeralfs	J	D	F	O	JDFO
alfisols	xeralfs	palexeralfs	haplic palexeralfs	J	D	F	P	JDFP
alfisols	xeralfs	palexeralfs	mollic palexeralfs	J	D	F	Q	JDFQ
alfisols	xeralfs	palexeralfs	typic palexeralfs	J	D	F	R	JDFR
alfisols	xeralfs	haploxeralfs	lithic mollic haploxeralfs	J	D	G	A	JDGA
alfisols	xeralfs	haploxeralfs	lithic ruptic-inceptic haploxeralfs	J	D	G	B	JDGB
alfisols	xeralfs	haploxeralfs	lithic haploxeralfs	J	D	G	C	JDGC
alfisols	xeralfs	haploxeralfs	vertic haploxeralfs	J	D	G	D	JDGD
alfisols	xeralfs	haploxeralfs	aquandic haploxeralfs	J	D	G	E	JDGE
alfisols	xeralfs	haploxeralfs	andic haploxeralfs	J	D	G	F	JDGF
alfisols	xeralfs	haploxeralfs	vitrandic haploxeralfs	J	D	G	G	JDGG
alfisols	xeralfs	haploxeralfs	fragiaquic haploxeralfs	J	D	G	H	JDGH
alfisols	xeralfs	haploxeralfs	aquultic haploxeralfs	J	D	G	I	JDGI
alfisols	xeralfs	haploxeralfs	aquic haploxeralfs	J	D	G	J	JDGJ
alfisols	xeralfs	haploxeralfs	natric haploxeralfs	J	D	G	K	JDGK
alfisols	xeralfs	haploxeralfs	fragic haploxeralfs	J	D	G	L	JDGL
alfisols	xeralfs	haploxeralfs	lamellic haploxeralfs	J	D	G	M	JDGM
alfisols	xeralfs	haploxeralfs	psammentic haploxeralfs	J	D	G	N	JDGN
alfisols	xeralfs	haploxeralfs	plinthic haploxeralfs	J	D	G	O	JDGO
alfisols	xeralfs	haploxeralfs	calcic haploxeralfs	J	D	G	P	JDGP
alfisols	xeralfs	haploxeralfs	inceptic haploxeralfs	J	D	G	Q	JDGQ
alfisols	xeralfs	haploxeralfs	ultic haploxeralfs	J	D	G	R	JDGR
alfisols	xeralfs	haploxeralfs	mollic haploxeralfs	J	D	G	S	JDGS
alfisols	xeralfs	haploxeralfs	typic haploxeralfs	J	D	G	T	JDGT
alfisols	udalfs	natrudalfs	vertic natrudalfs	J	E	A	A	JEAA
alfisols	udalfs	natrudalfs	glossaquic natrudalfs	J	E	A	B	JEAB
alfisols	udalfs	natrudalfs	aquic natrudalfs	J	E	A	C	JEAC
alfisols	udalfs	natrudalfs	typic natrudalfs	J	E	A	D	JEAD
alfisols	udalfs	ferrudalfs	aquic ferrudalfs	J	E	B	A	JEBA
alfisols	udalfs	ferrudalfs	typic ferrudalfs	J	E	B	B	JEBB
alfisols	udalfs	fraglossudalfs	andic fraglossudalfs	J	E	C	A	JECA
alfisols	udalfs	fraglossudalfs	vitrandic fraglossudalfs	J	E	C	B	JECB
alfisols	udalfs	fraglossudalfs	aquic fraglossudalfs	J	E	C	C	JECC
alfisols	udalfs	fraglossudalfs	oxyaquic fraglossudalfs	J	E	C	D	JECD
alfisols	udalfs	fraglossudalfs	typic fraglossudalfs	J	E	C	E	JECE
alfisols	udalfs	fragiudalfs	andic fragiudalfs	J	E	D	A	JEDA
alfisols	udalfs	fragiudalfs	vitrandic fragiudalfs	J	E	D	B	JEDB
alfisols	udalfs	fragiudalfs	aquic fragiudalfs	J	E	D	C	JEDC
alfisols	udalfs	fragiudalfs	oxyaquic fragiudalfs	J	E	D	D	JEDD
alfisols	udalfs	fragiudalfs	typic fragiudalfs	J	E	D	E	JEDE
alfisols	udalfs	kandiudalfs	plinthaquic kandiudalfs	J	E	E	A	JEEA

alfisols	udalfs	kandiudalfs	aquic kandiudalfs	J	E	E	B	JEEB
alfisols	udalfs	kandiudalfs	oxyaquic kandiudalfs	J	E	E	C	JEEC
alfisols	udalfs	kandiudalfs	arenic plinthic kandiudalfs	J	E	E	D	JEED
alfisols	udalfs	kandiudalfs	grossarenic plinthic kandiudalfs	J	E	E	E	JEEE
alfisols	udalfs	kandiudalfs	arenic kandiudalfs	J	E	E	F	JEEF
alfisols	udalfs	kandiudalfs	grossarenic kandiudalfs	J	E	E	G	JEEG
alfisols	udalfs	kandiudalfs	plinthic kandiudalfs	J	E	E	H	JEEH
alfisols	udalfs	kandiudalfs	rhodic kandiudalfs	J	E	E	I	JEEI
alfisols	udalfs	kandiudalfs	mollic kandiudalfs	J	E	E	J	JEEJ
alfisols	udalfs	kandiudalfs	typic kandiudalfs	J	E	E	K	JEEK
alfisols	udalfs	kanhapludalfs	lithic kanhapludalfs	J	E	F	A	JEFA
alfisols	udalfs	kanhapludalfs	aquic kanhapludalfs	J	E	F	B	JEFB
alfisols	udalfs	kanhapludalfs	oxyaquic kanhapludalfs	J	E	F	C	JEFC
alfisols	udalfs	kanhapludalfs	rhodic kanhapludalfs	J	E	F	D	JEFD
alfisols	udalfs	kanhapludalfs	typic kanhapludalfs	J	E	F	E	JEFE
alfisols	udalfs	paleudalfs	vertic paleudalfs	J	E	G	A	JEGA
alfisols	udalfs	paleudalfs	andic paleudalfs	J	E	G	B	JEGB
alfisols	udalfs	paleudalfs	vitrandic paleudalfs	J	E	G	C	JEGC
alfisols	udalfs	paleudalfs	anthraquic paleudalfs	J	E	G	D	JEGD
alfisols	udalfs	paleudalfs	fragiaquic paleudalfs	J	E	G	E	JEGE
alfisols	udalfs	paleudalfs	plinthaquic paleudalfs	J	E	G	F	JEGF
alfisols	udalfs	paleudalfs	glossaquic paleudalfs	J	E	G	G	JEGG
alfisols	udalfs	paleudalfs	albaquic paleudalfs	J	E	G	H	JEGH
alfisols	udalfs	paleudalfs	aquic paleudalfs	J	E	G	I	JEGI
alfisols	udalfs	paleudalfs	oxyaquic paleudalfs	J	E	G	J	JEGJ
alfisols	udalfs	paleudalfs	fragic paleudalfs	J	E	G	K	JEGK
alfisols	udalfs	paleudalfs	arenic plinthic paleudalfs	J	E	G	L	JEGL
alfisols	udalfs	paleudalfs	grossarenic plinthic paleudalfs	J	E	G	M	JEGM
alfisols	udalfs	paleudalfs	lamellic paleudalfs	J	E	G	N	JEGN
alfisols	udalfs	paleudalfs	psammentic paleudalfs	J	E	G	O	JEGO
alfisols	udalfs	paleudalfs	arenic paleudalfs	J	E	G	P	JEGP
alfisols	udalfs	paleudalfs	grossarenic paleudalfs	J	E	G	Q	JEGQ
alfisols	udalfs	paleudalfs	plinthic paleudalfs	J	E	G	R	JEGR
alfisols	udalfs	paleudalfs	glossic paleudalfs	J	E	G	S	JEGS
alfisols	udalfs	paleudalfs	rhodic paleudalfs	J	E	G	T	JEGT
alfisols	udalfs	paleudalfs	mollic paleudalfs	J	E	G	U	JEGU
alfisols	udalfs	paleudalfs	typic paleudalfs	J	E	G	V	JEGV
alfisols	udalfs	rhodudalfs	typic rhodudalfs	J	E	H	A	JEHA
alfisols	udalfs	glossudalfs	aquertic glossudalfs	J	E	I	A	JEIA
alfisols	udalfs	glossudalfs	oxyaquic vertic glossudalfs	J	E	I	B	JEIB
alfisols	udalfs	glossudalfs	vertic glossudalfs	J	E	I	C	JEIC
alfisols	udalfs	glossudalfs	aquandic glossudalfs	J	E	I	D	JEID
alfisols	udalfs	glossudalfs	andic glossudalfs	J	E	I	E	JEIE
alfisols	udalfs	glossudalfs	vitrandic glossudalfs	J	E	I	F	JEIF
alfisols	udalfs	glossudalfs	fragiaquic glossudalfs	J	E	I	G	JEIG
alfisols	udalfs	glossudalfs	aquic arenic glossudalfs	J	E	I	H	JEIH
alfisols	udalfs	glossudalfs	aquic glossudalfs	J	E	I	I	JEII
alfisols	udalfs	glossudalfs	arenic oxyaquic glossudalfs	J	E	I	J	JEIJ
alfisols	udalfs	glossudalfs	oxyaquic glossudalfs	J	E	I	K	JEIK
alfisols	udalfs	glossudalfs	fragic glossudalfs	J	E	I	L	JEIL
alfisols	udalfs	glossudalfs	arenic glossudalfs	J	E	I	M	JEIM
alfisols	udalfs	glossudalfs	haplic glossudalfs	J	E	I	N	JEIN
alfisols	udalfs	glossudalfs	typic glossudalfs	J	E	I	O	JEIO
alfisols	udalfs	hapludalfs	lithic hapludalfs	J	E	J	A	JEJA
alfisols	udalfs	hapludalfs	aquertic chromic hapludalfs	J	E	J	B	JEJB
alfisols	udalfs	hapludalfs	aquertic hapludalfs	J	E	J	C	JEJC
alfisols	udalfs	hapludalfs	oxyaquic vertic hapludalfs	J	E	J	D	JEJD
alfisols	udalfs	hapludalfs	chromic vertic hapludalfs	J	E	J	E	JEJE
alfisols	udalfs	hapludalfs	vertic hapludalfs	J	E	J	F	JEJF
alfisols	udalfs	hapludalfs	andic hapludalfs	J	E	J	G	JEJG

alfisols	udalfs	hapludalfs	vitrandic hapludalfs	J	E	J	H	JEJH
alfisols	udalfs	hapludalfs	fragiaquic hapludalfs	J	E	J	I	JEJI
alfisols	udalfs	hapludalfs	fragic oxyaquic hapludalfs	J	E	J	J	JEJJ
alfisols	udalfs	hapludalfs	aquic arenic hapludalfs	J	E	J	K	JEJK
alfisols	udalfs	hapludalfs	arenic oxyaquic hapludalfs	J	E	J	L	JEJL
alfisols	udalfs	hapludalfs	anthraquic hapludalfs	J	E	J	M	JEJM
alfisols	udalfs	hapludalfs	albaquultic hapludalfs	J	E	J	N	JEJN
alfisols	udalfs	hapludalfs	albaquic hapludalfs	J	E	J	O	JEJO
alfisols	udalfs	hapludalfs	glossaquic hapludalfs	J	E	J	P	JEJP
alfisols	udalfs	hapludalfs	aquultic hapludalfs	J	E	J	Q	JEJQ
alfisols	udalfs	hapludalfs	aquollic hapludalfs	J	E	J	R	JEJR
alfisols	udalfs	hapludalfs	aquic hapludalfs	J	E	J	S	JEJS
alfisols	udalfs	hapludalfs	mollic oxyaquic hapludalfs	J	E	J	T	JEJT
alfisols	udalfs	hapludalfs	oxyaquic hapludalfs	J	E	J	U	JEJU
alfisols	udalfs	hapludalfs	fragic hapludalfs	J	E	J	V	JEJV
alfisols	udalfs	hapludalfs	lamellic hapludalfs	J	E	J	W	JEJW
alfisols	udalfs	hapludalfs	psammentic hapludalfs	J	E	J	X	JEJX
alfisols	udalfs	hapludalfs	arenic hapludalfs	J	E	J	Y	JEJY
alfisols	udalfs	hapludalfs	glossic hapludalfs	J	E	J	Z	JEJZ
alfisols	udalfs	hapludalfs	inceptic hapludalfs	J	E	J	Z	JEJZ
alfisols	udalfs	hapludalfs	ultic hapludalfs	J	E	J	Z	JEJZ
alfisols	udalfs	hapludalfs	mollic hapludalfs	J	E	J	Z	JEJZ
alfisols	udalfs	hapludalfs	typic hapludalfs	J	E	J	Z	JEJZ
inceptisols	aquepts	sulfaquepts	salidic sulfaquepts	K	A	A	A	KAAA
inceptisols	aquepts	sulfaquepts	hydraquentic sulfaquepts	K	A	A	B	KAAB
inceptisols	aquepts	sulfaquepts	typic sulfaquepts	K	A	A	C	KAAC
inceptisols	aquepts	petraquepts	histic placic petraquepts	K	A	B	A	KABA
inceptisols	aquepts	petraquepts	placic petraquepts	K	A	B	B	KABB
inceptisols	aquepts	petraquepts	plinthic petraquepts	K	A	B	C	KABC
inceptisols	aquepts	petraquepts	typic petraquepts	K	A	B	D	KABD
inceptisols	aquepts	halaquepts	vertic halaquepts	K	A	C	A	KACA
inceptisols	aquepts	halaquepts	aquandic halaquepts	K	A	C	B	KACB
inceptisols	aquepts	halaquepts	duric halaquepts	K	A	C	C	KACC
inceptisols	aquepts	halaquepts	aeric halaquepts	K	A	C	D	KACD
inceptisols	aquepts	halaquepts	typic halaquepts	K	A	C	E	KACE
inceptisols	aquepts	fragiaquepts	aeric fragiaquepts	K	A	D	A	KADA
inceptisols	aquepts	fragiaquepts	humic fragiaquepts	K	A	D	B	KADB
inceptisols	aquepts	fragiaquepts	typic fragiaquepts	K	A	D	C	KADC
inceptisols	aquepts	gelaquepts	lithic gelaquepts	K	A	E	A	KAEA
inceptisols	aquepts	gelaquepts	histic gelaquepts	K	A	E	B	KAEB
inceptisols	aquepts	gelaquepts	aquandic gelaquepts	K	A	E	C	KAEC
inceptisols	aquepts	gelaquepts	fluvaquentic gelaquepts	K	A	E	D	KAED
inceptisols	aquepts	gelaquepts	humic gelaquepts	K	A	E	E	KAEE
inceptisols	aquepts	gelaquepts	turbic gelaquepts	K	A	E	F	KAEF
inceptisols	aquepts	gelaquepts	typic gelaquepts	K	A	E	G	KAEG
inceptisols	aquepts	cryaquepts	sulfic cryaquepts	K	A	F	A	KAFA
inceptisols	aquepts	cryaquepts	histic lithic cryaquepts	K	A	F	B	KAFB
inceptisols	aquepts	cryaquepts	lithic cryaquepts	K	A	F	C	KAFC
inceptisols	aquepts	cryaquepts	vertic cryaquepts	K	A	F	D	KAFD
inceptisols	aquepts	cryaquepts	histic cryaquepts	K	A	F	E	KAFE
inceptisols	aquepts	cryaquepts	aquandic cryaquepts	K	A	F	F	KAFF
inceptisols	aquepts	cryaquepts	fluvaquentic cryaquepts	K	A	F	G	KAFG
inceptisols	aquepts	cryaquepts	aeric humic cryaquepts	K	A	F	H	KAFH
inceptisols	aquepts	cryaquepts	aeric cryaquepts	K	A	F	I	KAFI
inceptisols	aquepts	cryaquepts	humic cryaquepts	K	A	F	J	KAFJ
inceptisols	aquepts	cryaquepts	typic cryaquepts	K	A	F	K	KAFK
inceptisols	aquepts	vermaquepts	sodic vermaquepts	K	A	G	A	KAGA
inceptisols	aquepts	vermaquepts	typic vermaquepts	K	A	G	B	KAGB
inceptisols	aquepts	humaquepts	hydraquentic humaquepts	K	A	H	A	KAHA
inceptisols	aquepts	humaquepts	histic humaquepts	K	A	H	B	KAHB

inceptisols	aquepts	humaquepts	aquandic humaquepts	K	A	H	C	KAHC
inceptisols	aquepts	humaquepts	cumulic humaquepts	K	A	H	D	KAHD
inceptisols	aquepts	humaquepts	fluvaquentic humaquepts	K	A	H	E	KAHE
inceptisols	aquepts	humaquepts	aeric humaquepts	K	A	H	F	KAHF
inceptisols	aquepts	humaquepts	typic humaquepts	K	A	H	G	KAHG
inceptisols	aquepts	epiaquepts	vertic epiaquepts	K	A	I	A	KAIA
inceptisols	aquepts	epiaquepts	aquandic epiaquepts	K	A	I	B	KAIB
inceptisols	aquepts	epiaquepts	fluvaquentic epiaquepts	K	A	I	C	KAIC
inceptisols	aquepts	epiaquepts	fragic epiaquepts	K	A	I	D	KAID
inceptisols	aquepts	epiaquepts	aeric epiaquepts	K	A	I	E	KAIE
inceptisols	aquepts	epiaquepts	humic epiaquepts	K	A	I	F	KAIF
inceptisols	aquepts	epiaquepts	mollic epiaquepts	K	A	I	G	KAIG
inceptisols	aquepts	epiaquepts	typic epiaquepts	K	A	I	H	KAIH
inceptisols	aquepts	endoaquepts	sulfic endoaquepts	K	A	J	A	KAJA
inceptisols	aquepts	endoaquepts	lithic endoaquepts	K	A	J	B	KAJB
inceptisols	aquepts	endoaquepts	vertic endoaquepts	K	A	J	C	KAJC
inceptisols	aquepts	endoaquepts	aquandic endoaquepts	K	A	J	D	KAJD
inceptisols	aquepts	endoaquepts	fluventic endoaquepts	K	A	J	E	KAJE
inceptisols	aquepts	endoaquepts	fluvaquentic endoaquepts	K	A	J	F	KAJF
inceptisols	aquepts	endoaquepts	fragic endoaquepts	K	A	J	G	KAJG
inceptisols	aquepts	endoaquepts	aeric endoaquepts	K	A	J	H	KAJH
inceptisols	aquepts	endoaquepts	humic endoaquepts	K	A	J	I	KAJI
inceptisols	aquepts	endoaquepts	mollic endoaquepts	K	A	J	J	KAJJ
inceptisols	aquepts	endoaquepts	typic endoaquepts	K	A	J	K	KAJK
inceptisols	gelepts	humigelepts	lithic humigelepts	K	B	A	A	KBAA
inceptisols	gelepts	humigelepts	andic humigelepts	K	B	A	B	KBAB
inceptisols	gelepts	humigelepts	aquic humigelepts	K	B	A	C	KBAC
inceptisols	gelepts	humigelepts	oxyaquic humigelepts	K	B	A	D	KBAD
inceptisols	gelepts	humigelepts	fluventic humigelepts	K	B	A	E	KBAE
inceptisols	gelepts	humigelepts	turbic humigelepts	K	B	A	F	KBAF
inceptisols	gelepts	humigelepts	eutric humigelepts	K	B	A	G	KBAG
inceptisols	gelepts	humigelepts	typic humigelepts	K	B	A	H	KBAH
inceptisols	gelepts	dystrogelepts	lithic dystrogelepts	K	B	B	A	KBBA
inceptisols	gelepts	dystrogelepts	andic dystrogelepts	K	B	B	B	KBBB
inceptisols	gelepts	dystrogelepts	aquic dystrogelepts	K	B	B	C	KBBC
inceptisols	gelepts	dystrogelepts	fluventic dystrogelepts	K	B	B	D	KBBD
inceptisols	gelepts	dystrogelepts	turbic dystrogelepts	K	B	B	E	KBBE
inceptisols	gelepts	dystrogelepts	typic dystrogelepts	K	B	B	F	KBBF
inceptisols	gelepts	haplogelepts	lithic haplogelepts	K	B	C	A	KBCA
inceptisols	gelepts	haplogelepts	andic haplogelepts	K	B	C	B	KBCB
inceptisols	gelepts	haplogelepts	aquic haplogelepts	K	B	C	C	KBCC
inceptisols	gelepts	haplogelepts	fluventic haplogelepts	K	B	C	D	KBCD
inceptisols	gelepts	haplogelepts	turbic haplogelepts	K	B	C	E	KBCE
inceptisols	gelepts	haplogelepts	typic haplogelepts	K	B	C	F	KBCF
inceptisols	cryepts	humicryepts	lithic humicryepts	K	C	A	A	KCAA
inceptisols	cryepts	humicryepts	aquandic humicryepts	K	C	A	B	KCAB
inceptisols	cryepts	humicryepts	haploxerandic humicryepts	K	C	A	C	KCAC
inceptisols	cryepts	humicryepts	vitrixerandic humicryepts	K	C	A	D	KCAD
inceptisols	cryepts	humicryepts	andic humicryepts	K	C	A	E	KCAE
inceptisols	cryepts	humicryepts	vitrandic humicryepts	K	C	A	F	KCAF
inceptisols	cryepts	humicryepts	fluvaquentic humicryepts	K	C	A	G	KCAG
inceptisols	cryepts	humicryepts	aquic humicryepts	K	C	A	H	KCAH
inceptisols	cryepts	humicryepts	oxyaquic humicryepts	K	C	A	I	KCAI
inceptisols	cryepts	humicryepts	lamellic humicryepts	K	C	A	J	KCAJ
inceptisols	cryepts	humicryepts	fluventic humicryepts	K	C	A	K	KCAK
inceptisols	cryepts	humicryepts	spodic humicryepts	K	C	A	L	KCAL
inceptisols	cryepts	humicryepts	xeric humicryepts	K	C	A	M	KCAM
inceptisols	cryepts	humicryepts	eutric humicryepts	K	C	A	N	KCAN
inceptisols	cryepts	humicryepts	typic humicryepts	K	C	A	O	KCAO
inceptisols	cryepts	calcicryepts	lithic calcicryepts	K	C	B	A	KCBA

inceptisols	cryepts	calcicryepts	oxyaquic calcicryepts	K	C	B	B	KCBB
inceptisols	cryepts	calcicryepts	xeric calcicryepts	K	C	B	C	KCBC
inceptisols	cryepts	calcicryepts	ustic calcicryepts	K	C	B	D	KCBD
inceptisols	cryepts	calcicryepts	typic calcicryepts	K	C	B	E	KCBE
inceptisols	cryepts	dystrocryepts	lithic dystrocryepts	K	C	C	A	KCCA
inceptisols	cryepts	dystrocryepts	aquandic dystrocryepts	K	C	C	B	KCCB
inceptisols	cryepts	dystrocryepts	haploxerandic dystrocryepts	K	C	C	C	KCCC
inceptisols	cryepts	dystrocryepts	vitrixerandic dystrocryepts	K	C	C	D	KCCD
inceptisols	cryepts	dystrocryepts	andic dystrocryepts	K	C	C	E	KCCE
inceptisols	cryepts	dystrocryepts	vitrandic dystrocryepts	K	C	C	F	KCCF
inceptisols	cryepts	dystrocryepts	fluvaquentic dystrocryepts	K	C	C	G	KCCG
inceptisols	cryepts	dystrocryepts	folistic dystrocryepts	K	C	C	H	KCCH
inceptisols	cryepts	dystrocryepts	aquic dystrocryepts	K	C	C	I	KCCI
inceptisols	cryepts	dystrocryepts	oxyaquic dystrocryepts	K	C	C	J	KCCJ
inceptisols	cryepts	dystrocryepts	lamellic dystrocryepts	K	C	C	K	KCCK
inceptisols	cryepts	dystrocryepts	fluventic dystrocryepts	K	C	C	L	KCCL
inceptisols	cryepts	dystrocryepts	spodic dystrocryepts	K	C	C	M	KCCM
inceptisols	cryepts	dystrocryepts	xeric dystrocryepts	K	C	C	N	KCCN
inceptisols	cryepts	dystrocryepts	ustic dystrocryepts	K	C	C	O	KCCO
inceptisols	cryepts	dystrocryepts	eutric dystrocryepts	K	C	C	P	KCCP
inceptisols	cryepts	dystrocryepts	typic dystrocryepts	K	C	C	Q	KCCQ
inceptisols	cryepts	haplocryepts	lithic haplocryepts	K	C	D	A	KCDA
inceptisols	cryepts	haplocryepts	aquandic haplocryepts	K	C	D	B	KCDB
inceptisols	cryepts	haplocryepts	haploxerandic haplocryepts	K	C	D	C	KCDC
inceptisols	cryepts	haplocryepts	vitrixerandic haplocryepts	K	C	D	D	KCDD
inceptisols	cryepts	haplocryepts	haplustandic haplocryepts	K	C	D	E	KCDE
inceptisols	cryepts	haplocryepts	ustivitrandic haplocryepts	K	C	D	F	KCDF
inceptisols	cryepts	haplocryepts	andic haplocryepts	K	C	D	G	KCDG
inceptisols	cryepts	haplocryepts	vitrandic haplocryepts	K	C	D	H	KCDH
inceptisols	cryepts	haplocryepts	fluvaquentic haplocryepts	K	C	D	I	KCDI
inceptisols	cryepts	haplocryepts	aquic haplocryepts	K	C	D	J	KCDJ
inceptisols	cryepts	haplocryepts	oxyaquic haplocryepts	K	C	D	K	KCDK
inceptisols	cryepts	haplocryepts	lamellic haplocryepts	K	C	D	L	KCDL
inceptisols	cryepts	haplocryepts	fluventic haplocryepts	K	C	D	M	KCDM
inceptisols	cryepts	haplocryepts	calcic haplocryepts	K	C	D	N	KCDN
inceptisols	cryepts	haplocryepts	xeric haplocryepts	K	C	D	O	KCDO
inceptisols	cryepts	haplocryepts	ustic haplocryepts	K	C	D	P	KCDP
inceptisols	cryepts	haplocryepts	typic haplocryepts	K	C	D	Q	KCDQ
inceptisols	ustepts	durustepts	typic durustepts	K	D	A	A	KDAA
inceptisols	ustepts	calciustepts	lithic petrocalcic calciustepts	K	D	B	A	KDBA
inceptisols	ustepts	calciustepts	lithic calciustepts	K	D	B	B	KDBB
inceptisols	ustepts	calciustepts	torrertic calciustepts	K	D	B	C	KDBC
inceptisols	ustepts	calciustepts	vertic calciustepts	K	D	B	D	KDBD
inceptisols	ustepts	calciustepts	petrocalcic calciustepts	K	D	B	E	KDBE
inceptisols	ustepts	calciustepts	gypsic calciustepts	K	D	B	F	KDBF
inceptisols	ustepts	calciustepts	aquic calciustepts	K	D	B	G	KDBG
inceptisols	ustepts	calciustepts	aridic calciustepts	K	D	B	H	KDBH
inceptisols	ustepts	calciustepts	udic calciustepts	K	D	B	I	KDBI
inceptisols	ustepts	calciustepts	typic calciustepts	K	D	B	J	KDBJ
inceptisols	ustepts	humustepts	lithic humustepts	K	D	C	A	KDCA
inceptisols	ustepts	humustepts	andic humustepts	K	D	C	B	KDCB
inceptisols	ustepts	humustepts	vitrandic humustepts	K	D	C	C	KDCC
inceptisols	ustepts	humustepts	oxyaquic humustepts	K	D	C	D	KDCD
inceptisols	ustepts	humustepts	oxic humustepts	K	D	C	E	KDCE
inceptisols	ustepts	humustepts	aridic humustepts	K	D	C	F	KDCF
inceptisols	ustepts	humustepts	typic humustepts	K	D	C	G	KDCG
inceptisols	ustepts	dystrustepts	lithic dystrustepts	K	D	D	A	KDDA
inceptisols	ustepts	dystrustepts	torrertic dystrustepts	K	D	D	B	KDDB
inceptisols	ustepts	dystrustepts	vertic dystrustepts	K	D	D	C	KDDC
inceptisols	ustepts	dystrustepts	andic dystrustepts	K	D	D	D	KDDD

inceptisols	ustepts	dystrustepts	vitrandic dystrustepts	K	D	D	E	KDDE
inceptisols	ustepts	dystrustepts	aquic dystrustepts	K	D	D	F	KDDF
inceptisols	ustepts	dystrustepts	fluventic dystrustepts	K	D	D	G	KDDG
inceptisols	ustepts	dystrustepts	aridic dystrustepts	K	D	D	H	KDDH
inceptisols	ustepts	dystrustepts	oxic dystrustepts	K	D	D	I	KDDI
inceptisols	ustepts	dystrustepts	humic dystrustepts	K	D	D	J	KDDJ
inceptisols	ustepts	dystrustepts	typic dystrustepts	K	D	D	K	KDDK
inceptisols	ustepts	haplustepts	aridic lithic haplustepts	K	D	E	A	KDEA
inceptisols	ustepts	haplustepts	lithic haplustepts	K	D	E	B	KDEB
inceptisols	ustepts	haplustepts	udertic haplustepts	K	D	E	C	KDEC
inceptisols	ustepts	haplustepts	torrertic haplustepts	K	D	E	D	KDED
inceptisols	ustepts	haplustepts	vertic haplustepts	K	D	E	E	KDEE
inceptisols	ustepts	haplustepts	andic haplustepts	K	D	E	F	KDEF
inceptisols	ustepts	haplustepts	vitrandic haplustepts	K	D	E	G	KDEG
inceptisols	ustepts	haplustepts	anthraquic haplustepts	K	D	E	H	KDEH
inceptisols	ustepts	haplustepts	aquic haplustepts	K	D	E	I	KDEI
inceptisols	ustepts	haplustepts	oxyaquic haplustepts	K	D	E	J	KDEJ
inceptisols	ustepts	haplustepts	oxic haplustepts	K	D	E	K	KDEK
inceptisols	ustepts	haplustepts	lamellic haplustepts	K	D	E	L	KDEL
inceptisols	ustepts	haplustepts	torrifluventic haplustepts	K	D	E	M	KDEM
inceptisols	ustepts	haplustepts	udifluventic haplustepts	K	D	E	N	KDEN
inceptisols	ustepts	haplustepts	fluventic haplustepts	K	D	E	O	KDEO
inceptisols	ustepts	haplustepts	gypsic haplustepts	K	D	E	P	KDEP
inceptisols	ustepts	haplustepts	haplocalcidic haplustepts	K	D	E	Q	KDEQ
inceptisols	ustepts	haplustepts	calcic udic haplustepts	K	D	E	R	KDER
inceptisols	ustepts	haplustepts	calcic haplustepts	K	D	E	S	KDES
inceptisols	ustepts	haplustepts	aridic haplustepts	K	D	E	T	KDET
inceptisols	ustepts	haplustepts	dystric haplustepts	K	D	E	U	KDEU
inceptisols	ustepts	haplustepts	udic haplustepts	K	D	E	V	KDEV
inceptisols	ustepts	haplustepts	typic haplustepts	K	D	E	W	KDEW
inceptisols	xerepts	durixerepts	aquandic durixerepts	K	E	A	A	KEAA
inceptisols	xerepts	durixerepts	andic durixerepts	K	E	A	B	KEAB
inceptisols	xerepts	durixerepts	vitrandic durixerepts	K	E	A	C	KEAC
inceptisols	xerepts	durixerepts	aquic durixerepts	K	E	A	D	KEAD
inceptisols	xerepts	durixerepts	entic durixerepts	K	E	A	E	KEAE
inceptisols	xerepts	durixerepts	typic durixerepts	K	E	A	F	KEAF
inceptisols	xerepts	fragixerepts	andic fragixerepts	K	E	B	A	KEBA
inceptisols	xerepts	fragixerepts	vitrandic fragixerepts	K	E	B	B	KEBB
inceptisols	xerepts	fragixerepts	aquic fragixerepts	K	E	B	C	KEBC
inceptisols	xerepts	fragixerepts	humic fragixerepts	K	E	B	D	KEBD
inceptisols	xerepts	fragixerepts	typic fragixerepts	K	E	B	E	KEBE
inceptisols	xerepts	humixerepts	lithic humixerepts	K	E	C	A	KECA
inceptisols	xerepts	humixerepts	aquandic humixerepts	K	E	C	B	KECB
inceptisols	xerepts	humixerepts	andic humixerepts	K	E	C	C	KECC
inceptisols	xerepts	humixerepts	vitrandic humixerepts	K	E	C	D	KECD
inceptisols	xerepts	humixerepts	aquic humixerepts	K	E	C	E	KECE
inceptisols	xerepts	humixerepts	oxyaquic humixerepts	K	E	C	F	KECF
inceptisols	xerepts	humixerepts	cumulic humixerepts	K	E	C	G	KECG
inceptisols	xerepts	humixerepts	fluventic humixerepts	K	E	C	H	KECH
inceptisols	xerepts	humixerepts	pachic humixerepts	K	E	C	I	KECI
inceptisols	xerepts	humixerepts	entic humixerepts	K	E	C	J	KECJ
inceptisols	xerepts	humixerepts	typic humixerepts	K	E	C	K	KECK
inceptisols	xerepts	calcixerepts	lithic calcixerepts	K	E	D	A	KEDA
inceptisols	xerepts	calcixerepts	vertic calcixerepts	K	E	D	B	KEDB
inceptisols	xerepts	calcixerepts	petrocalcic calcixerepts	K	E	D	C	KEDC
inceptisols	xerepts	calcixerepts	sodic calcixerepts	K	E	D	D	KEDD
inceptisols	xerepts	calcixerepts	vitrandic calcixerepts	K	E	D	E	KEDE
inceptisols	xerepts	calcixerepts	aquic calcixerepts	K	E	D	F	KEDF
inceptisols	xerepts	calcixerepts	typic calcixerepts	K	E	D	G	KEDG
inceptisols	xerepts	dystroxerepts	humic lithic dystroxerepts	K	E	E	A	KEEA

inceptisols	xerepts	dystroxerepts	lithic dystroxerepts	K	E	E	B	KEEB
inceptisols	xerepts	dystroxerepts	aquandic dystroxerepts	K	E	E	C	KEEC
inceptisols	xerepts	dystroxerepts	andic dystroxerepts	K	E	E	D	KEED
inceptisols	xerepts	dystroxerepts	vitrandic dystroxerepts	K	E	E	E	KEEE
inceptisols	xerepts	dystroxerepts	fragiaquic dystroxerepts	K	E	E	F	KEEF
inceptisols	xerepts	dystroxerepts	fluvaquentic dystroxerepts	K	E	E	G	KEEG
inceptisols	xerepts	dystroxerepts	aquic dystroxerepts	K	E	E	H	KEEH
inceptisols	xerepts	dystroxerepts	oxyaquic dystroxerepts	K	E	E	I	KEEI
inceptisols	xerepts	dystroxerepts	fragic dystroxerepts	K	E	E	J	KEEJ
inceptisols	xerepts	dystroxerepts	fluventic humic dystroxerepts	K	E	E	K	KEEK
inceptisols	xerepts	dystroxerepts	fluventic dystroxerepts	K	E	E	L	KEEL
inceptisols	xerepts	dystroxerepts	humic dystroxerepts	K	E	E	M	KEEM
inceptisols	xerepts	dystroxerepts	typic dystroxerepts	K	E	E	N	KEEN
inceptisols	xerepts	haploxerepts	humic lithic haploxerepts	K	E	F	A	KEFA
inceptisols	xerepts	haploxerepts	lithic haploxerepts	K	E	F	B	KEFB
inceptisols	xerepts	haploxerepts	vertic haploxerepts	K	E	F	C	KEFC
inceptisols	xerepts	haploxerepts	aquandic haploxerepts	K	E	F	D	KEFD
inceptisols	xerepts	haploxerepts	andic oxyaquic haploxerepts	K	E	F	E	KEFE
inceptisols	xerepts	haploxerepts	andic haploxerepts	K	E	F	F	KEFF
inceptisols	xerepts	haploxerepts	oxyaquic vitrandic haploxerepts	K	E	F	G	KEFG
inceptisols	xerepts	haploxerepts	vitrandic haploxerepts	K	E	F	H	KEFH
inceptisols	xerepts	haploxerepts	gypsic haploxerepts	K	E	F	I	KEFI
inceptisols	xerepts	haploxerepts	aquic haploxerepts	K	E	F	J	KEFJ
inceptisols	xerepts	haploxerepts	lamellic haploxerepts	K	E	F	K	KEFK
inceptisols	xerepts	haploxerepts	fragic haploxerepts	K	E	F	L	KEFL
inceptisols	xerepts	haploxerepts	fluventic haploxerepts	K	E	F	M	KEFM
inceptisols	xerepts	haploxerepts	calcic haploxerepts	K	E	F	N	KEFN
inceptisols	xerepts	haploxerepts	humic haploxerepts	K	E	F	O	KEFO
inceptisols	xerepts	haploxerepts	typic haploxerepts	K	E	F	P	KEFP
inceptisols	udepts	sulfudepts	typic sulfudepts	K	F	A	A	KFAA
inceptisols	udepts	durudepts	aquandic durudepts	K	F	B	A	KFBA
inceptisols	udepts	durudepts	andic durudepts	K	F	B	B	KFBB
inceptisols	udepts	durudepts	vitrandic durudepts	K	F	B	C	KFBC
inceptisols	udepts	durudepts	aquic durudepts	K	F	B	D	KFBD
inceptisols	udepts	durudepts	typic durudepts	K	F	B	E	KFBE
inceptisols	udepts	fragiudepts	andic fragiudepts	K	F	C	A	KFCA
inceptisols	udepts	fragiudepts	vitrandic fragiudepts	K	F	C	B	KFCB
inceptisols	udepts	fragiudepts	aquic fragiudepts	K	F	C	C	KFCC
inceptisols	udepts	fragiudepts	humic fragiudepts	K	F	C	D	KFCD
inceptisols	udepts	fragiudepts	typic fragiudepts	K	F	C	E	KFCE
inceptisols	udepts	humudepts	lithic humudepts	K	F	D	A	KFDA
inceptisols	udepts	humudepts	vertic humudepts	K	F	D	B	KFDB
inceptisols	udepts	humudepts	aquandic humudepts	K	F	D	C	KFDC
inceptisols	udepts	humudepts	andic oxyaquic humudepts	K	F	D	D	KFDD
inceptisols	udepts	humudepts	andic humudepts	K	F	D	E	KFDE
inceptisols	udepts	humudepts	vitrandic humudepts	K	F	D	F	KFDF
inceptisols	udepts	humudepts	fluvaquentic humudepts	K	F	D	G	KFDG
inceptisols	udepts	humudepts	aquic humudepts	K	F	D	H	KFDH
inceptisols	udepts	humudepts	oxyaquic humudepts	K	F	D	I	KFDI
inceptisols	udepts	humudepts	psammentic humudepts	K	F	D	J	KFDJ
inceptisols	udepts	humudepts	oxic humudepts	K	F	D	K	KFDK
inceptisols	udepts	humudepts	cumulic humudepts	K	F	D	L	KFDL
inceptisols	udepts	humudepts	fluventic humudepts	K	F	D	M	KFDM
inceptisols	udepts	humudepts	pachic humudepts	K	F	D	N	KFDN
inceptisols	udepts	humudepts	eutric humudepts	K	F	D	O	KFDO
inceptisols	udepts	humudepts	entic humudepts	K	F	D	P	KFDP
inceptisols	udepts	humudepts	typic humudepts	K	F	D	Q	KFDQ
inceptisols	udepts	eutrudepts	humic lithic eutrudepts	K	F	E	A	KFEA
inceptisols	udepts	eutrudepts	lithic eutrudepts	K	F	E	B	KFEB
inceptisols	udepts	eutrudepts	aquertic eutrudepts	K	F	E	C	KFEC

inceptisols	udepts	eutrudepts	vertic eutrudepts	K	F	E	D	KFED
inceptisols	udepts	eutrudepts	andic eutrudepts	K	F	E	E	KFEE
inceptisols	udepts	eutrudepts	vitrandic eutrudepts	K	F	E	F	KFEF
inceptisols	udepts	eutrudepts	anthraquic eutrudepts	K	F	E	G	KFEG
inceptisols	udepts	eutrudepts	fragiaquic eutrudepts	K	F	E	H	KFEH
inceptisols	udepts	eutrudepts	fluvaquentic eutrudepts	K	F	E	I	KFEI
inceptisols	udepts	eutrudepts	aquic dystric eutrudepts	K	F	E	J	KFEJ
inceptisols	udepts	eutrudepts	aquic eutrudepts	K	F	E	K	KFEK
inceptisols	udepts	eutrudepts	oxyaquic eutrudepts	K	F	E	L	KFEL
inceptisols	udepts	eutrudepts	fragic eutrudepts	K	F	E	M	KFEM
inceptisols	udepts	eutrudepts	lamellic eutrudepts	K	F	E	N	KFEN
inceptisols	udepts	eutrudepts	dystric fluventic eutrudepts	K	F	E	O	KFEO
inceptisols	udepts	eutrudepts	fluventic eutrudepts	K	F	E	P	KFEP
inceptisols	udepts	eutrudepts	arenic eutrudepts	K	F	E	Q	KFEQ
inceptisols	udepts	eutrudepts	dystric eutrudepts	K	F	E	R	KFER
inceptisols	udepts	eutrudepts	rendollic eutrudepts	K	F	E	S	KFES
inceptisols	udepts	eutrudepts	ruptic-alfic eutrudepts	K	F	E	T	KFET
inceptisols	udepts	eutrudepts	humic eutrudepts	K	F	E	U	KFEU
inceptisols	udepts	eutrudepts	typic eutrudepts	K	F	E	V	KFEV
inceptisols	udepts	dystrudepts	humic lithic dystrudepts	K	F	F	A	KFFA
inceptisols	udepts	dystrudepts	lithic dystrudepts	K	F	F	B	KFFB
inceptisols	udepts	dystrudepts	vertic dystrudepts	K	F	F	C	KFFC
inceptisols	udepts	dystrudepts	aquandic dystrudepts	K	F	F	D	KFFD
inceptisols	udepts	dystrudepts	andic oxyaquic dystrudepts	K	F	F	E	KFFE
inceptisols	udepts	dystrudepts	andic dystrudepts	K	F	F	F	KFFF
inceptisols	udepts	dystrudepts	vitrandic dystrudepts	K	F	F	G	KFFG
inceptisols	udepts	dystrudepts	fragiaquic dystrudepts	K	F	F	H	KFFH
inceptisols	udepts	dystrudepts	fluvaquentic dystrudepts	K	F	F	I	KFFI
inceptisols	udepts	dystrudepts	aquic humic dystrudepts	K	F	F	J	KFFJ
inceptisols	udepts	dystrudepts	aquic dystrudepts	K	F	F	K	KFFK
inceptisols	udepts	dystrudepts	oxyaquic dystrudepts	K	F	F	L	KFFL
inceptisols	udepts	dystrudepts	fragic dystrudepts	K	F	F	M	KFFM
inceptisols	udepts	dystrudepts	lamellic dystrudepts	K	F	F	N	KFFN
inceptisols	udepts	dystrudepts	humic psammentic dystrudepts	K	F	F	O	KFFO
inceptisols	udepts	dystrudepts	fluventic humic dystrudepts	K	F	F	P	KFFP
inceptisols	udepts	dystrudepts	fluventic dystrudepts	K	F	F	Q	KFFQ
inceptisols	udepts	dystrudepts	spodic dystrudepts	K	F	F	R	KFFR
inceptisols	udepts	dystrudepts	oxic dystrudepts	K	F	F	S	KFFS
inceptisols	udepts	dystrudepts	ruptic-alfic dystrudepts	K	F	F	T	KFFT
inceptisols	udepts	dystrudepts	ruptic-ultic dystrudepts	K	F	F	U	KFFU
inceptisols	udepts	dystrudepts	humic dystrudepts	K	F	F	V	KFFV
inceptisols	udepts	dystrudepts	typic dystrudepts	K	F	F	W	KFFW
entisols	wassents	frasiwassents	hydric frasiwassents	L	A	A	A	LAAA
entisols	wassents	frasiwassents	lithic frasiwassents	L	A	A	B	LAAB
entisols	wassents	frasiwassents	psammentic frasiwassents	L	A	A	C	LAAC
entisols	wassents	frasiwassents	thapto-histic frasiwassents	L	A	A	D	LAAD
entisols	wassents	frasiwassents	fluventic frasiwassents	L	A	A	E	LAAE
entisols	wassents	frasiwassents	aeric frasiwassents	L	A	A	F	LAAF
entisols	wassents	frasiwassents	typic frasiwassents	L	A	A	G	LAAG
entisols	wassents	psammowassents	sulfic psammowassents	L	A	B	A	LABA
entisols	wassents	psammowassents	lithic psammowassents	L	A	B	B	LABB
entisols	wassents	psammowassents	fluventic psammowassents	L	A	B	C	LABC
entisols	wassents	psammowassents	aeric psammowassents	L	A	B	D	LABD
entisols	wassents	psammowassents	typic psammowassents	L	A	B	E	LABE
entisols	wassents	sulfiwassents	lithic sulfiwassents	L	A	C	A	LACA
entisols	wassents	sulfiwassents	haplic sulfiwassents	L	A	C	B	LACB
entisols	wassents	sulfiwassents	thapto-histic sulfiwassents	L	A	C	C	LACC
entisols	wassents	sulfiwassents	fluventic sulfiwassents	L	A	C	D	LACD
entisols	wassents	sulfiwassents	aeric sulfiwassents	L	A	C	E	LACE
entisols	wassents	sulfiwassents	typic sulfiwassents	L	A	C	F	LACF

entisols	wassents	hydrowassents	sulfic hydrowassents	L	A	D	A	LADA
entisols	wassents	hydrowassents	grossic hydrowassents	L	A	D	B	LADB
entisols	wassents	hydrowassents	lithic hydrowassents	L	A	D	C	LADC
entisols	wassents	hydrowassents	thapto-histic hydrowassents	L	A	D	D	LADD
entisols	wassents	hydrowassents	typic hydrowassents	L	A	D	E	LADE
entisols	wassents	fluviwassents	sulfic fluviwassents	L	A	E	A	LAEA
entisols	wassents	fluviwassents	lithic fluviwassents	L	A	E	B	LAEB
entisols	wassents	fluviwassents	thapto-histic fluviwassents	L	A	E	C	LAEC
entisols	wassents	fluviwassents	aeric fluviwassents	L	A	E	D	LAED
entisols	wassents	fluviwassents	typic fluviwassents	L	A	E	E	LAEE
entisols	wassents	haplowassents	sulfic haplowassents	L	A	F	A	LAFA
entisols	wassents	haplowassents	lithic haplowassents	L	A	F	B	LAFB
entisols	wassents	haplowassents	aeric haplowassents	L	A	F	C	LAFC
entisols	wassents	haplowassents	typic haplowassents	L	A	F	D	LAFD
entisols	aquents	sulfaquents	haplic sulfaquents	L	B	A	A	LBAA
entisols	aquents	sulfaquents	histic sulfaquents	L	B	A	B	LBAB
entisols	aquents	sulfaquents	thapto-histic sulfaquents	L	B	A	C	LBAC
entisols	aquents	sulfaquents	typic sulfaquents	L	B	A	D	LBAD
entisols	aquents	hydraquents	sulfic hydraquents	L	B	B	A	LBBA
entisols	aquents	hydraquents	sodic hydraquents	L	B	B	B	LBBB
entisols	aquents	hydraquents	thapto-histic hydraquents	L	B	B	C	LBBC
entisols	aquents	hydraquents	typic hydraquents	L	B	B	D	LBBD
entisols	aquents	gelaquents	typic gelaquents	L	B	C	A	LBCA
entisols	aquents	cryaquents	aquandic cryaquents	L	B	D	A	LBDA
entisols	aquents	cryaquents	typic cryaquents	L	B	D	B	LBDB
entisols	aquents	psammaquents	lithic psammaquents	L	B	E	A	LBEA
entisols	aquents	psammaquents	sodic psammaquents	L	B	E	B	LBEB
entisols	aquents	psammaquents	spodic psammaquents	L	B	E	C	LBEC
entisols	aquents	psammaquents	humaqueptic psammaquents	L	B	E	D	LBED
entisols	aquents	psammaquents	mollic psammaquents	L	B	E	E	LBEE
entisols	aquents	psammaquents	typic psammaquents	L	B	E	F	LBEF
entisols	aquents	fluvaquents	sulfic fluvaquents	L	B	F	A	LBFA
entisols	aquents	fluvaquents	vertic fluvaquents	L	B	F	B	LBFB
entisols	aquents	fluvaquents	thapto-histic fluvaquents	L	B	F	C	LBFC
entisols	aquents	fluvaquents	aquandic fluvaquents	L	B	F	D	LBFD
entisols	aquents	fluvaquents	aeric fluvaquents	L	B	F	E	LBFE
entisols	aquents	fluvaquents	humaqueptic fluvaquents	L	B	F	F	LBFF
entisols	aquents	fluvaquents	mollic fluvaquents	L	B	F	G	LBFG
entisols	aquents	fluvaquents	typic fluvaquents	L	B	F	H	LBFH
entisols	aquents	epiaquents	aeric epiaquents	L	B	G	A	LBGA
entisols	aquents	epiaquents	humaqueptic epiaquents	L	B	G	B	LBGB
entisols	aquents	epiaquents	mollic epiaquents	L	B	G	C	LBGC
entisols	aquents	epiaquents	typic epiaquents	L	B	G	D	LBGD
entisols	aquents	endoaquents	sulfic endoaquents	L	B	H	A	LBHA
entisols	aquents	endoaquents	lithic endoaquents	L	B	H	B	LBHB
entisols	aquents	endoaquents	sodic endoaquents	L	B	H	C	LBHC
entisols	aquents	endoaquents	aeric endoaquents	L	B	H	D	LBHD
entisols	aquents	endoaquents	humaqueptic endoaquents	L	B	H	E	LBHE
entisols	aquents	endoaquents	mollic endoaquents	L	B	H	F	LBHF
entisols	aquents	endoaquents	typic endoaquents	L	B	H	G	LBHG
entisols	psamments	cryopsamments	lithic cryopsamments	L	C	A	A	LCAA
entisols	psamments	cryopsamments	aquic cryopsamments	L	C	A	B	LCAB
entisols	psamments	cryopsamments	oxyaquic cryopsamments	L	C	A	C	LCAC
entisols	psamments	cryopsamments	vitrandic cryopsamments	L	C	A	D	LCAD
entisols	psamments	cryopsamments	spodic cryopsamments	L	C	A	E	LCAE
entisols	psamments	cryopsamments	lamellic cryopsamments	L	C	A	F	LCAF
entisols	psamments	cryopsamments	typic cryopsamments	L	C	A	G	LCAG
entisols	psamments	torripsamments	lithic torripsamments	L	C	B	A	LCBA
entisols	psamments	torripsamments	oxyaquic torripsamments	L	C	B	B	LCBB
entisols	psamments	torripsamments	vitrandic torripsamments	L	C	B	C	LCBC

entisols	psamments	torripsamments	haploduridic torripsamments	L	C	B	D	LCBD
entisols	psamments	torripsamments	ustic torripsamments	L	C	B	E	LCBE
entisols	psamments	torripsamments	xeric torripsamments	L	C	B	F	LCBF
entisols	psamments	torripsamments	rhodic torripsamments	L	C	B	G	LCBG
entisols	psamments	torripsamments	typic torripsamments	L	C	B	H	LCBH
entisols	psamments	quartzipsamments	lithic quartzipsamments	L	C	C	A	LCCA
entisols	psamments	quartzipsamments	aquodic quartzipsamments	L	C	C	B	LCCB
entisols	psamments	quartzipsamments	aquic quartzipsamments	L	C	C	C	LCCC
entisols	psamments	quartzipsamments	oxyaquic quartzipsamments	L	C	C	D	LCCD
entisols	psamments	quartzipsamments	ustoxic quartzipsamments	L	C	C	E	LCCE
entisols	psamments	quartzipsamments	udoxic quartzipsamments	L	C	C	F	LCCF
entisols	psamments	quartzipsamments	plinthic quartzipsamments	L	C	C	G	LCCG
entisols	psamments	quartzipsamments	lamellic ustic quartzipsamments	L	C	C	H	LCCH
entisols	psamments	quartzipsamments	lamellic quartzipsamments	L	C	C	I	LCCI
entisols	psamments	quartzipsamments	ustic quartzipsamments	L	C	C	J	LCCJ
entisols	psamments	quartzipsamments	xeric quartzipsamments	L	C	C	K	LCCK
entisols	psamments	quartzipsamments	spodic quartzipsamments	L	C	C	L	LCCL
entisols	psamments	quartzipsamments	typic quartzipsamments	L	C	C	M	LCCM
entisols	psamments	ustipsamments	lithic ustipsamments	L	C	D	A	LCDA
entisols	psamments	ustipsamments	aquic ustipsamments	L	C	D	B	LCDB
entisols	psamments	ustipsamments	oxyaquic ustipsamments	L	C	D	C	LCDC
entisols	psamments	ustipsamments	aridic ustipsamments	L	C	D	D	LCDD
entisols	psamments	ustipsamments	lamellic ustipsamments	L	C	D	E	LCDE
entisols	psamments	ustipsamments	rhodic ustipsamments	L	C	D	F	LCDF
entisols	psamments	ustipsamments	typic ustipsamments	L	C	D	G	LCDG
entisols	psamments	xeropsamments	lithic xeropsamments	L	C	E	A	LCEA
entisols	psamments	xeropsamments	aquic durinodic xeropsamments	L	C	E	B	LCEB
entisols	psamments	xeropsamments	aquic xeropsamments	L	C	E	C	LCEC
entisols	psamments	xeropsamments	oxyaquic xeropsamments	L	C	E	D	LCED
entisols	psamments	xeropsamments	vitrandic xeropsamments	L	C	E	E	LCEE
entisols	psamments	xeropsamments	durinodic xeropsamments	L	C	E	F	LCEF
entisols	psamments	xeropsamments	lamellic xeropsamments	L	C	E	G	LCEG
entisols	psamments	xeropsamments	dystric xeropsamments	L	C	E	H	LCEH
entisols	psamments	xeropsamments	typic xeropsamments	L	C	E	I	LCEI
entisols	psamments	udipsamments	lithic udipsamments	L	C	F	A	LCFA
entisols	psamments	udipsamments	aquic udipsamments	L	C	F	B	LCFB
entisols	psamments	udipsamments	oxyaquic udipsamments	L	C	F	C	LCFC
entisols	psamments	udipsamments	spodic udipsamments	L	C	F	D	LCFD
entisols	psamments	udipsamments	lamellic udipsamments	L	C	F	E	LCFE
entisols	psamments	udipsamments	haploplaggic udipsamments	L	C	F	F	LCFF
entisols	psamments	udipsamments	typic udipsamments	L	C	F	G	LCFG
entisols	fluvents	gelifluvents	aquic gelifluvents	L	D	A	A	LDAA
entisols	fluvents	gelifluvents	typic gelifluvents	L	D	A	B	LDAB
entisols	fluvents	cryofluvents	andic cryofluvents	L	D	B	A	LDBA
entisols	fluvents	cryofluvents	vitrandic cryofluvents	L	D	B	B	LDBB
entisols	fluvents	cryofluvents	aquic cryofluvents	L	D	B	C	LDBC
entisols	fluvents	cryofluvents	oxyaquic cryofluvents	L	D	B	D	LDBD
entisols	fluvents	cryofluvents	mollic cryofluvents	L	D	B	E	LDBE
entisols	fluvents	cryofluvents	typic cryofluvents	L	D	B	F	LDBF
entisols	fluvents	xerofluvents	vertic xerofluvents	L	D	C	A	LDCA
entisols	fluvents	xerofluvents	aquandic xerofluvents	L	D	C	B	LDCB
entisols	fluvents	xerofluvents	andic xerofluvents	L	D	C	C	LDCC
entisols	fluvents	xerofluvents	vitrandic xerofluvents	L	D	C	D	LDCD
entisols	fluvents	xerofluvents	aquic xerofluvents	L	D	C	E	LDCE
entisols	fluvents	xerofluvents	oxyaquic xerofluvents	L	D	C	F	LDCF
entisols	fluvents	xerofluvents	durinodic xerofluvents	L	D	C	G	LDCG
entisols	fluvents	xerofluvents	mollic xerofluvents	L	D	C	H	LDCH
entisols	fluvents	xerofluvents	typic xerofluvents	L	D	C	I	LDCI
entisols	fluvents	ustifluvents	aquertic ustifluvents	L	D	D	A	LDDA
entisols	fluvents	ustifluvents	torrertic ustifluvents	L	D	D	B	LDDB

entisols	fluvents	ustifluvents	vertic ustifluvents	L	D	D	C	LDDC
entisols	fluvents	ustifluvents	anthraquic ustifluvents	L	D	D	D	LDDD
entisols	fluvents	ustifluvents	aquic ustifluvents	L	D	D	E	LDDE
entisols	fluvents	ustifluvents	oxyaquic ustifluvents	L	D	D	F	LDDF
entisols	fluvents	ustifluvents	aridic ustifluvents	L	D	D	G	LDDG
entisols	fluvents	ustifluvents	udic ustifluvents	L	D	D	H	LDDH
entisols	fluvents	ustifluvents	mollic ustifluvents	L	D	D	I	LDDI
entisols	fluvents	ustifluvents	typic ustifluvents	L	D	D	J	LDDJ
entisols	fluvents	torrifluvents	ustertic torrifluvents	L	D	E	A	LDEA
entisols	fluvents	torrifluvents	vertic torrifluvents	L	D	E	B	LDEB
entisols	fluvents	torrifluvents	vitrixerandic torrifluvents	L	D	E	C	LDEC
entisols	fluvents	torrifluvents	vitrandic torrifluvents	L	D	E	D	LDED
entisols	fluvents	torrifluvents	aquic torrifluvents	L	D	E	E	LDEE
entisols	fluvents	torrifluvents	oxyaquic torrifluvents	L	D	E	F	LDEF
entisols	fluvents	torrifluvents	duric xeric torrifluvents	L	D	E	G	LDEG
entisols	fluvents	torrifluvents	duric torrifluvents	L	D	E	H	LDEH
entisols	fluvents	torrifluvents	ustic torrifluvents	L	D	E	I	LDEI
entisols	fluvents	torrifluvents	xeric torrifluvents	L	D	E	J	LDEJ
entisols	fluvents	torrifluvents	anthropic torrifluvents	L	D	E	K	LDEK
entisols	fluvents	torrifluvents	typic torrifluvents	L	D	E	L	LDEL
entisols	fluvents	udifluvents	aquertic udifluvents	L	D	F	A	LDFA
entisols	fluvents	udifluvents	vertic udifluvents	L	D	F	B	LDFB
entisols	fluvents	udifluvents	andic udifluvents	L	D	F	C	LDFC
entisols	fluvents	udifluvents	vitrandic udifluvents	L	D	F	D	LDFD
entisols	fluvents	udifluvents	aquic udifluvents	L	D	F	E	LDFE
entisols	fluvents	udifluvents	oxyaquic udifluvents	L	D	F	F	LDFF
entisols	fluvents	udifluvents	mollic udifluvents	L	D	F	G	LDFG
entisols	fluvents	udifluvents	typic udifluvents	L	D	F	H	LDFH
entisols	orthents	gelorthents	aquic gelorthents	L	E	A	A	LEAA
entisols	orthents	gelorthents	oxyaquic gelorthents	L	E	A	B	LEAB
entisols	orthents	gelorthents	typic gelorthents	L	E	A	C	LEAC
entisols	orthents	cryorthents	lithic cryorthents	L	E	B	A	LEBA
entisols	orthents	cryorthents	vitrandic cryorthents	L	E	B	B	LEBB
entisols	orthents	cryorthents	aquic cryorthents	L	E	B	C	LEBC
entisols	orthents	cryorthents	oxyaquic cryorthents	L	E	B	D	LEBD
entisols	orthents	cryorthents	lamellic cryorthents	L	E	B	E	LEBE
entisols	orthents	cryorthents	typic cryorthents	L	E	B	F	LEBF
entisols	orthents	torriorthents	lithic ustic torriorthents	L	E	C	A	LECA
entisols	orthents	torriorthents	lithic xeric torriorthents	L	E	C	B	LECB
entisols	orthents	torriorthents	lithic torriorthents	L	E	C	C	LECC
entisols	orthents	torriorthents	xerertic torriorthents	L	E	C	D	LECD
entisols	orthents	torriorthents	ustertic torriorthents	L	E	C	E	LECE
entisols	orthents	torriorthents	vertic torriorthents	L	E	C	F	LECF
entisols	orthents	torriorthents	anthraltic torriorthents	L	E	C	G	LECG
entisols	orthents	torriorthents	vitrandic torriorthents	L	E	C	H	LECH
entisols	orthents	torriorthents	aquic torriorthents	L	E	C	I	LECI
entisols	orthents	torriorthents	oxyaquic torriorthents	L	E	C	J	LECJ
entisols	orthents	torriorthents	duric torriorthents	L	E	C	K	LECK
entisols	orthents	torriorthents	ustic torriorthents	L	E	C	L	LECL
entisols	orthents	torriorthents	xeric torriorthents	L	E	C	M	LECM
entisols	orthents	torriorthents	typic torriorthents	L	E	C	N	LECN
entisols	orthents	xerorthents	lithic xerorthents	L	E	D	A	LEDA
entisols	orthents	xerorthents	anthraltic sodic xerorthents	L	E	D	B	LEDB
entisols	orthents	xerorthents	anthraltic xerorthents	L	E	D	C	LEDC
entisols	orthents	xerorthents	vitrandic xerorthents	L	E	D	D	LEDD
entisols	orthents	xerorthents	aquic xerorthents	L	E	D	E	LEDE
entisols	orthents	xerorthents	oxyaquic xerorthents	L	E	D	F	LEDF
entisols	orthents	xerorthents	durinodic xerorthents	L	E	D	G	LEDG
entisols	orthents	xerorthents	dystric xerorthents	L	E	D	H	LEDH
entisols	orthents	xerorthents	typic xerorthents	L	E	D	I	LEDI

entisols	orthents	ustorthents	aridic lithic ustorthents	L	E	E	A	LEEA
entisols	orthents	ustorthents	lithic ustorthents	L	E	E	B	LEEB
entisols	orthents	ustorthents	torrertic ustorthents	L	E	E	C	LEEC
entisols	orthents	ustorthents	vertic ustorthents	L	E	E	D	LEED
entisols	orthents	ustorthents	anthraquic ustorthents	L	E	E	E	LEEE
entisols	orthents	ustorthents	anthrodensic ustorthents	L	E	E	F	LEEF
entisols	orthents	ustorthents	anthroportic ustorthents	L	E	E	G	LEEG
entisols	orthents	ustorthents	aquic ustorthents	L	E	E	H	LEEH
entisols	orthents	ustorthents	oxyaquic ustorthents	L	E	E	I	LEEI
entisols	orthents	ustorthents	durinodic ustorthents	L	E	E	J	LEEJ
entisols	orthents	ustorthents	vitritorrandic ustorthents	L	E	E	K	LEEK
entisols	orthents	ustorthents	vitrandic ustorthents	L	E	E	L	LEEL
entisols	orthents	ustorthents	aridic ustorthents	L	E	E	M	LEEM
entisols	orthents	ustorthents	udic ustorthents	L	E	E	N	LEEN
entisols	orthents	ustorthents	vermic ustorthents	L	E	E	O	LEEO
entisols	orthents	ustorthents	typic ustorthents	L	E	E	P	LEEP
entisols	orthents	udorthents	lithic udorthents	L	E	F	A	LEFA
entisols	orthents	udorthents	anthrodensic sodic udorthents	L	E	F	B	LEFB
entisols	orthents	udorthents	anthrodensic udorthents	L	E	F	C	LEFC
entisols	orthents	udorthents	anthropic udorthents	L	E	F	D	LEFD
entisols	orthents	udorthents	anthroportic udorthents	L	E	F	E	LEFE
entisols	orthents	udorthents	vitrandic udorthents	L	E	F	F	LEFF
entisols	orthents	udorthents	aquic udorthents	L	E	F	G	LEFG
entisols	orthents	udorthents	oxyaquic udorthents	L	E	F	H	LEFH
entisols	orthents	udorthents	vermic udorthents	L	E	F	I	LEFI
entisols	orthents	udorthents	typic udorthents	L	E	F	J	LEFJ