

# Package ‘soilDB’

May 16, 2021

**Type** Package

**Title** Soil Database Interface

**Version** 2.6.2

**Author** Dylan Beaudette [aut],  
Jay Skovlin [aut],  
Stephen Roecker [aut],  
Andrew Brown [aut, cre]

**Maintainer** Andrew Brown <andrew.g.brown@usda.gov>

**Description** A collection of functions for reading data from USDA-NCSS soil databases.

**License** GPL (>= 3)

**LazyLoad** yes

**Depends** R (>= 3.5.0)

**Imports** aqp, grDevices, graphics, stats, utils, plyr, xml2, sp,  
reshape2, raster, curl, methods, data.table, DBI

**Suggests** rgdal, jsonlite, httr, sf, rgeos, rvest, odbc, RSQLite,  
testthat, gridExtra, ggplot2, viridisLite, mapview, rasterVis

**Repository** CRAN

**URL** <http://ncss-tech.github.io/AQP/>

**BugReports** <https://github.com/ncss-tech/soilDB/issues>

**RoxygenNote** 7.1.1

**NeedsCompilation** no

**Date/Publication** 2021-05-16 14:30:02 UTC

## R topics documented:

soilDB-package . . . . .	3
.dump_NASIS_table . . . . .	4
.get_NASIS_table_key_by_name . . . . .	4
.get_NASIS_table_name_by_purpose . . . . .	5
createStaticNASIS . . . . .	6

dbConnectNASIS . . . . .	7
dbQueryNASIS . . . . .	7
estimateColorMixture . . . . .	8
estimateSTR . . . . .	8
fetchGDB . . . . .	10
fetchHenry . . . . .	11
fetchKSSL . . . . .	13
fetchNASIS . . . . .	15
fetchNASISLabData . . . . .	17
fetchNASISWebReport . . . . .	18
fetchOSD . . . . .	21
fetchPедonPC . . . . .	24
fetchRaCA . . . . .	25
fetchSCAN . . . . .	26
fetchSDA . . . . .	28
fetchSDA_spatial . . . . .	31
fetchSoilGrids . . . . .	33
fetchVegdata . . . . .	34
filter_geochem . . . . .	35
format_SQL_in_statement . . . . .	36
getHzErrorsNASIS . . . . .	37
get_colors_from_NASIS_db . . . . .	38
get_colors_from_pedon_db . . . . .	38
get_comonth_from_NASIS_db . . . . .	39
get_component_data_from_NASIS_db . . . . .	40
get_cosoilmoist_from_NASIS . . . . .	41
get_EDIT_ecoclass_by_geoUnit . . . . .	43
get_extended_data_from_NASIS_db . . . . .	44
get_extended_data_from_pedon_db . . . . .	45
get_hz_data_from_NASIS_db . . . . .	46
get_hz_data_from_pedon_db . . . . .	47
get_lablayer_data_from_NASIS_db . . . . .	48
get_labpedon_data_from_NASIS_db . . . . .	49
get_NOAA_GHCND . . . . .	50
get_NOAA_stations_nearXY . . . . .	51
get_OSD_JSON . . . . .	52
get_SDA_hydric . . . . .	53
get_SDA_interpretation . . . . .	53
get_SDA_muaggatt . . . . .	69
get_SDA_pmgrouppname . . . . .	70
get_SDA_property . . . . .	70
get_site_data_from_NASIS_db . . . . .	73
get_site_data_from_pedon_db . . . . .	74
get_soilseries_from_NASIS . . . . .	75
get_text_notes_from_NASIS_db . . . . .	75
get_veg_data_from_NASIS_db . . . . .	76
get_veg_from_AK_Site . . . . .	77
get_veg_from_MT_veg_db . . . . .	78

get_veg_from_NPS_PLOTS_db . . . . .	79
get_veg_other_from_MT_veg_db . . . . .	79
get_veg_species_from_MT_veg_db . . . . .	80
ISSR800.wcs . . . . .	81
KSSL_VG_model . . . . .	82
loafercreek . . . . .	83
local_NASIS_defined . . . . .	84
makeChunks . . . . .	85
make_EDIT_service_URL . . . . .	86
mapunit_geom_by_ll_bbox . . . . .	88
mix_and_clean_colors . . . . .	89
mukey.wcs . . . . .	90
NASIS_table_column_keys . . . . .	91
OSDquery . . . . .	91
parseWebReport . . . . .	93
processSDA_WKT . . . . .	94
ROSETTA . . . . .	95
SCAN_SNOTEL_metadata . . . . .	97
SDA_query . . . . .	98
SDA_spatialQuery . . . . .	100
seriesExtent . . . . .	104
siblings . . . . .	105
simplifyColorData . . . . .	107
simplifyFragmentData . . . . .	108
SoilWeb_spatial_query . . . . .	109
STRplot . . . . .	110
taxaExtent . . . . .	111
unicode . . . . .	116
us_ss_timeline . . . . .	118
waterDayYear . . . . .	119
WCS_details . . . . .	120

<b>Index</b>	<b>121</b>
--------------	------------

---

soilDB-package	<i>Soil Database Interface</i>
----------------	--------------------------------

---

## Description

This package provides methods for extracting soils information from local PedonPC and AK Site databases (MS Access format), local NASIS databases (MS SQL Server), and the SDA web service. Currently USDA-NCSS data sources are supported, however, there are plans to develop interfaces to outside systems such as the Global Soil Mapping project.

## Author(s)

J.M. Skovlin, D.E. Beaudette, S.M. Roecker, A.G. Brown

**See Also**

[fetchPedinPC](#), [fetchNASIS](#), [SDA\\_query](#), [loafercreek](#)

---

`.dump_NASIS_table`      *Method for "dumping" contents of an entire NASIS table*

---

**Description**

Method for "dumping" contents of an entire NASIS table

**Usage**

```
.dump_NASIS_table(table_name, dsn = NULL)
```

**Arguments**

<code>table_name</code>	Character name of table.
<code>dsn</code>	Optional: path to SQLite database containing NASIS table structure; Default: NULL

**Value**

A data.frame or other result of `DBI::dbGetQuery`

---

`.get_NASIS_table_key_by_name`  
*Get a NASIS table key by type and table name*

---

**Description**

Get a NASIS table key by type and table name

**Usage**

```
.get_NASIS_table_key_by_name(  
  tables,  
  keycol = c("all", "fkey", "pkeyref", "pkey")  
)
```

**Arguments**

<code>tables</code>	character vector of table names
<code>keycol</code>	One of: "fkey" the foreign key; "pkeyref" the primary key referenced by the foreign key, or "pkey" the primary key.

**Value**

The key column name for the specified table name

**Examples**

```
## Not run:  
.get_NASIS_table_key_by_name(c("site", "phorizon_View_1", "not_a_table"))  
  
## End(Not run)
```

---

```
.get_NASIS_table_name_by_purpose
```

*Return NASIS 7 Physical Table Names*

---

**Description**

Draft method generalizing concepts of NASIS 7 data model to group tables by "purpose." Not intended for typical user-level, and not exported at this point. Most of our more complex queries rely on tables from one or more purposes, so individual higher-level functions might call a function like this to identify the relevant tables from some data source.

**Usage**

```
.get_NASIS_table_name_by_purpose(  
  purpose = c("metadata", "lookup", "site", "pedon", "transect", "component",  
             "vegetation", "project", "techsoilservice", "area", "soilseries", "legend",  
             "mapunit", "datamapunit"),  
  SS = FALSE  
)
```

**Arguments**

purpose	character. One or more of: "metadata", "lookup", "site", "pedon", "transect", "datamapunit", "component", "vegetation"
SS	append "_View_1" on appropriate tables? Default: FALSE

**Value**

character vector of table names

**See Also**

`createStaticNASIS`

**Examples**

```
## Not run:
# get the "site" table names
.get_NASIS_table_name_by_purpose("site")

# get the pedon table names
.get_NASIS_table_name_by_purpose("pedon", SS = TRUE)

# metadata and lookup not affected by SS argument, but site and pedon are
.get_NASIS_table_name_by_purpose(c("metadata", "lookup",
                                "site", "pedon"), SS = TRUE)

## End(Not run)
```

---

```
createStaticNASIS      Create a memory or file-based instance of NASIS database
```

---

**Description**

Create a memory or file-based instance of NASIS database for selected tables.

**Usage**

```
createStaticNASIS(
  tables = NULL,
  new_names = NULL,
  SS = TRUE,
  dsn = NULL,
  output_path = NULL,
  verbose = FALSE
)
```

**Arguments**

tables	Character vector of target tables. Default: NULL is whatever tables are listed by <code>DBI::dbListTables</code> for the connection typ being used.
new_names	Optional: new table names (should match length of vector of matching tables in dsn)
SS	Logical. Include "selected set" tables (ending with suffix <code>"_View_1"</code> ). Default: TRUE
dsn	Optional: path to SQLite database containing NASIS table structure; Default: NULL
output_path	Optional: path to new/existing SQLite database to write tables to. Default: NULL returns table results as named list.
verbose	Show error messages from attempts to dump individual tables? Default FALSE

**Value**

A named list of results from calling dbQueryNASIS for all columns in each NASIS table.

---

dbConnectNASIS	<i>Create a connection to a local NASIS database</i>
----------------	------------------------------------------------------

---

**Description**

Create a connection to a local NASIS database with DBI

**Usage**

```
dbConnectNASIS(dsn = NULL)
```

**Arguments**

dsn	Optional: path to SQLite database containing NASIS table structure; Default: NULL
-----	-----------------------------------------------------------------------------------

**Value**

A DBIConnection object, as returned by DBI::dbConnect(). If dsn is a DBIConnection, the attribute isUserDefined of the result is set to TRUE. If the DBIConnection is created by the internal NASIS connection process, isUserDefined is set to FALSE.

---

dbQueryNASIS	<i>Send queries to a NASIS DBIConnection</i>
--------------	----------------------------------------------

---

**Description**

Send queries to a NASIS DBIConnection

**Usage**

```
dbQueryNASIS(conn, q, close = TRUE, ...)
```

**Arguments**

conn	A DBIConnection object, as returned by DBI::dbConnect().
q	A statement to execute using DBI::dbGetQuery; or a (named) vector containing multiple statements to evaluate separately
close	Close connection after query? Default: TRUE
...	Additional arguments to DBI::dbGetQuery

**Value**

Result of DBI::dbGetQuery

---

estimateColorMixture	<i>Estimate color mixtures using weighted average of CIELAB color coordinates</i>
----------------------	-----------------------------------------------------------------------------------

---

### Description

Estimate color mixtures using weighted average of CIELAB color coordinates

### Usage

```
estimateColorMixture(x, wt = "pct", backTransform = FALSE)
```

### Arguments

x	data.frame, typically from NASIS containing at least CIE LAB ('L', 'A', 'B') and some kind of weight
wt	fractional weights, usually area of hz face
backTransform	logical, should the mixed sRGB representation of soil color be transformed to closest Munsell chips? This is performed by aqp::rgb2Munsell default: FALSE

### Value

A data.frame containing estimated color mixture

### Note

See [mixMunsell](#) for a more realistic (but slower) simulation of subtractive mixing of pigments.

### Author(s)

D.E. Beaudette

---

estimateSTR	<i>Estimate Soil Temperature Regime</i>
-------------	-----------------------------------------

---

### Description

Estimate soil temperature regime (STR) based on mean annual soil temperature (MAST), mean summer temperature (MSST), mean winter soil temperature (MWST), presence of O horizons, saturated conditions, and presence of permafrost. Several assumptions are made when O horizon or saturation are undefined.



**Usage**

```
estimateSTR(  
  mast,  
  mean.summer,  
  mean.winter,  
  O.hz = NA,  
  saturated = NA,  
  permafrost = FALSE  
)
```

**Arguments**

mast	vector of mean annual soil temperature (deg C)
mean.summer	vector of mean summer soil temperature (deg C)
mean.winter	vector of mean winter soil temperature (deg C)
O.hz	logical vector of O horizon presence / absence
saturated	logical vector of seasonal saturation
permafrost	logical vector of permafrost presence / absence

**Details**

[Soil Temperature Regime Evaluation Tutorial](#)

**Value**

Vector of soil temperature regimes.

**Author(s)**

D.E. Beaudette

**References**

Soil Survey Staff. 2015. Illustrated guide to soil taxonomy. U.S. Department of Agriculture, Natural Resources Conservation Service, National Soil Survey Center, Lincoln, Nebraska.

**See Also**

[STRplot](#)

**Examples**

```
# simple example  
estimateSTR(mast=17, mean.summer = 22, mean.winter = 12)
```

---

 fetchGDB

*Load and Flatten Data from SSURGO file geodatabases*


---

### Description

Functions to load and flatten commonly used tables and from SSURGO file geodatabases, and create soil profile collection objects (SPC).

### Usage

```
fetchGDB(
  dsn = "gNATSGO_CONUS.gdb",
  WHERE = NULL,
  childs = TRUE,
  droplevels = TRUE,
  stringsAsFactors = TRUE
)
```

### Arguments

dsn	data source name (interpretation varies by driver - for some drivers, dsn is a file name, but may also be a folder, or contain the name and access credentials of a database); in case of GeoJSON, dsn may be the character string holding the geojson data. It can also be an open database connection.
WHERE	text string formatted as an SQL WHERE clause (default: FALSE)
childs	logical; if FALSE parent material and geomorphic child tables are not flattened and appended
droplevels	logical: indicating whether to drop unused levels in classifying factors. This is useful when a class has large number of unused classes, which can waste space in tables and figures.
stringsAsFactors	logical: should character vectors be converted to factors? This argument is passed to the uncode() function. It does not convert those vectors that have set outside of uncode() (i.e. hard coded). The 'factory-fresh' default is TRUE, but this can be changed by setting options(stringsAsFactors = FALSE)

### Details

These functions return data from SSURGO file geodatabases with the use of a simple text string that formatted as an SQL WHERE clause (e.g. WHERE = "areasymbol = 'IN001' ". Any columns within the target table can be specified (except for fetchGDB() currently, which only targets the legend with the WHERE clause).

### Value

A data.frame or SoilProfileCollection object.

**Author(s)**

Stephen Roecker

**Examples**

```
## replace `dsn` with path to your own geodatabase (SSURGO OR gNATSGO)
##
##
## download CONUS gNATSGO from here:
## https://www.nrcs.usda.gov/wps/portal/nrcs/detail/soils/survey/geo/?cid=nrcseprd1464625
##
##
# dsn <- "D:/geodata/soils/gNATSGO_CONUS.gdb"

# le <- get_legend_from_GDB(dsn = dsn, WHERE = "areasybol LIKE '%")

# mu <- get_mapunit_from_GDB(dsn = dsn, WHERE = "muname LIKE 'Miami%")

# co <- get_component_from_GDB(dsn, WHERE = "compname = 'Miami'
#                               AND majcompflag = 'Yes'", childs = FALSE)

# f_in_GDB <- fetchGDB(WHERE = "areasybol LIKE 'IN%")
```

---

 fetchHenry

*Download Data from the Henry Mount Soil Temperature and Water Database*

---

**Description**

This function is a front-end to the REST query functionality of the Henry Mount Soil Temperature and Water Database.

**Usage**

```
fetchHenry(
  what = "all",
  usersiteid = NULL,
  project = NULL,
  sso = NULL,
  gran = "day",
  start.date = NULL,
  stop.date = NULL,
  pad.missing.days = TRUE,
```

```

    soiltemp.summaries = TRUE
  )

```

### Arguments

what	type of data to return: 'sensors': sensor metadata only   'soiltemp': sensor metadata + soil temperature data   'soilVWC': sensor metadata + soil moisture data   'airtemp': sensor metadata + air temperature data   'waterlevel': sensor metadata + water level data   'all': sensor metadata + all sensor data
usersiteid	(optional) filter results using a NASIS user site ID
project	(optional) filter results using a project ID
sso	(optional) filter results using a soil survey office code
gran	data granularity: "day", "week", "month", "year"; returned data are averages
start.date	(optional) starting date filter
stop.date	(optional) ending date filter
pad.missing.days	should missing data ("day" granularity) be filled with NA? see details
soiltemp.summaries	should soil temperature ("day" granularity only) be summarized? see details

### Details

Filling missing days with NA is useful for computing and index of how complete the data are, and for estimating (mostly) unbiased MAST and seasonal mean soil temperatures. Summaries are computed by first averaging over Julian day, then averaging over all days of the year (MAST) or just those days that occur within "summer" or "winter". This approach makes it possible to estimate summaries in the presence of missing data. The quality of summaries should be weighted by the number of "functional years" (number of years with non-missing data after combining data by Julian day) and "complete years" (number of years of data with  $\geq 365$  days of non-missing data).

See:

- [Henry Mount Soil Climate Database](#)
- [fetchHenry Tutorial](#)

### Value

a list containing:

sensors	a SpatialPointsDataFrame object containing site-level information
soiltemp	a data.frame object containing soil temperature timeseries data
soilVWC	a data.frame object containing soil moisture timeseries data
airtemp	a data.frame object containing air temperature timeseries data
waterlevel	a data.frame object containing water level timeseries data

### Note

This function and the back-end database are very much a work in progress.

**Author(s)**

D.E. Beaudette

**See Also**[fetchSCAN](#)


---

fetchKSSL	<i>Fetch KSSL Data</i>
-----------	------------------------

---

**Description**

Download soil characterization and morphologic data via BBOX, MLRA, or soil series name query, from the KSSL database.

**Usage**

```
fetchKSSL(
  series = NA,
  bbox = NA,
  mlra = NA,
  pedlabsampnum = NA,
  pedon_id = NA,
  pedon_key = NA,
  returnMorphologicData = FALSE,
  returnGeochemicalData = FALSE,
  simplifyColors = FALSE,
  progress = TRUE
)
```

**Arguments**

series	vector of soil series names, case insensitive
bbox	a single bounding box in WGS84 geographic coordinates e.g. c(-120, 37, -122, 38)
mlra	vector of MLRA IDs, e.g. "18" or "22A"
pedlabsampnum	vector of KSSL pedon lab sample number
pedon_id	vector of user pedon ID
pedon_key	vector of KSSL internal pedon ID
returnMorphologicData	logical, optionally request basic morphologic data, see details section
returnGeochemicalData	logical, optionally request geochemical, optical and XRD/thermal data, see details section
simplifyColors	logical, simplify colors (from morphologic data) and join with horizon data
progress	logical, optionally give progress when iterating over multiple requests

## Details

This is an experimental interface to a subset for the most commonly used data from a snapshot of KSSL (lab characterization) and NASIS (morphologic) data.

Series-queries are case insensitive. Series name is based on the "correlated as" field (from KSSL snapshot) when present. The "sampled as" classification was promoted to "correlated as" if the "correlated as" classification was missing.

When `returnMorphologicData` is TRUE, the resulting object is a list. The standard output from `fetchKSSL` (SoilProfileCollection object) is stored in the named element "SPC". The additional elements are basic morphologic data: soil color, rock fragment volume, pores, structure, and redoximorphic features. There is a 1:many relationship between the horizon data in "SPC" and the additional dataframes in `morph`. See examples for ideas on how to "flatten" these tables.

When `returnGeochemicalData` is TRUE, the resulting object is a list. The standard output from `fetchKSSL` (SoilProfileCollection object) is stored in the named element "SPC". The additional elements are geochemical and mineralogy analysis tables, specifically: geochemical/elemental analyses "geochem", optical mineralogy "optical", and X-ray diffraction / thermal "xrd\_thermal". `returnGeochemicalData` will include additional dataframes `geochem`, `optical`, and `xrd_thermal` in list result.

Setting `simplifyColors=TRUE` will automatically flatten the soil color data and join to horizon level attributes.

Function arguments (`series`, `mlra`, etc.) are fully vectorized except for `bbox`.

## Value

a SoilProfileCollection object when `returnMorphologicData` is FALSE, otherwise a list.

## Note

SoilWeb maintains a snapshot of these KSSL and NASIS data. The SoilWeb snapshot was developed using methods described here: <https://github.com/dylanbeaudette/process-kssl-snapshot>. Please use the link below for the live data.

## Author(s)

D.E. Beaudette and A.G. Brown

## References

<http://ncsslslabdatamart.sc.egov.usda.gov/>

## See Also

[fetchOSD](#)

**Examples**

```

if(requireNamespace("curl") &
  curl::has_internet()) {

  library(aqp)
  library(plyr)
  library(reshape2)

  # search by series name
  s <- fetchKSSL(series='auburn')

  # search by bounding-box
  # s <- fetchKSSL(bbox=c(-120, 37, -122, 38))

  # how many pedons
  length(s)

  # plot
  plotSPC(s, name='hzn_desgn', max.depth=150)

  ##
  ## morphologic data
  ##

  # get lab and morphologic data
  s <- fetchKSSL(series='auburn', returnMorphologicData = TRUE)

  # extract SPC
  pedons <- s$SPC

  ## automatically simplify color data
  s <- fetchKSSL(series='auburn', returnMorphologicData = TRUE, simplifyColors=TRUE)

  # check
  par(mar=c(0,0,0,0))
  plot(pedons, color='moist_soil_color', print.id=FALSE)

}

```

---

 fetchNASIS

---

*Fetch commonly used site/pedon/horizon or component data from NASIS.*


---

**Description**

Fetch commonly used site/pedon/horizon data or component from NASIS, returned as a SoilProfileCollection object.

**Usage**

```

fetchNASIS(
  from = "pedons",
  url = NULL,
  SS = TRUE,
  rmHzErrors = TRUE,
  nullFragAreZero = TRUE,
  soilColorState = "moist",
  lab = FALSE,
  fill = FALSE,
  stringsAsFactors = default.stringsAsFactors(),
  dsn = NULL
)

```

**Arguments**

from	determines what objects should be fetched? ('pedons'   'components'   'pedon_report')
url	string specifying the url for the NASIS pedon_report (default: NULL)
SS	fetch data from the currently loaded selected set in NASIS or from the entire local database (default: TRUE)
rmHzErrors	should pedons with horizon depth errors be removed from the results? (default: TRUE)
nullFragAreZero	should fragment volumes of NULL be interpreted as 0? (default: TRUE), see details
soilColorState	which colors should be used to generate the convenience field soil_color? ('moist' or 'dry')
lab	should the phlabresults child table be fetched with site/pedon/horizon data (default: FALSE)
fill	(fetchNASIS(from='components') only): include component records without horizon data in result? (default: FALSE)
stringsAsFactors	logical: should character vectors be converted to factors? This argument is passed to the uncode() function. It does not convert those vectors that have been set outside of uncode() (i.e. hard coded).
dsn	Optional: path to local SQLite database containing NASIS table structure; default: NULL

**Details**

This function imports data from NASIS into R as a `SoilProfileCollection` object. It "flattens" NASIS pedon and component tables, including their child tables, into several more easily manageable data frames. Primarily these functions access the local NASIS database using an ODBC connection. However using the `fetchNASIS()` argument `from = "pedon_report"`, data can be read from the NASIS Report 'fetchNASIS', as either a txt file or url. The primary purpose of



fetchNASIS(from = "pedon\_report") is to facilitate importing datasets larger than 8000+ pedons/components.

The value of nullFragmentsAreZero will have a significant impact on the rock fragment fractions returned by fetchNASIS. Set nullFragmentsAreZero = FALSE in those cases where there are many data-gaps and NULL rock fragment values should be interpreted as NULL. Set nullFragmentsAreZero = TRUE in those cases where NULL rock fragment values should be interpreted as 0.

This function attempts to do most of the boilerplate work when extracting site/pedon/horizon or component data from a local NASIS database. Pedons that are missing horizon data, or have errors in their horizonation are excluded from the returned object, however, their IDs are printed on the console. Pedons with combination horizons (e.g. B/C) are erroneously marked as errors due to the way in which they are stored in NASIS as two overlapping horizon records.

Tutorials:

- [fetchNASIS Pedons Tutorial](#)
- [fetchNASIS Components Tutorial](#)

### Value

A SoilProfileCollection object

### Author(s)

D. E. Beaudette, J. M. Skovlin, S.M. Roeker, A.G. Brown

---

fetchNASISLabData	<i>Fetch lab data used site/horizon data from a PedonPC database.</i>
-------------------	-----------------------------------------------------------------------

---

### Description

Fetch KSSL laboratory pedon/horizon layer data from a local NASIS database, return as a SoilProfileCollection object.

### Usage

```
fetchNASISLabData(SS = TRUE, dsn = NULL)
```

### Arguments

SS	fetch data from the currently loaded selected set in NASIS or from the entire local database (default: TRUE)#'
dsn	Optional: path to local SQLite database containing NASIS table structure; default: NULL

### Details

This function currently works only on Windows, and requires a 'nasis\_local' ODBC connection.

**Value**

a SoilProfileCollection object

**Author(s)**

J.M. Skovlin and D.E. Beaudette

**See Also**

[get\\_labpedon\\_data\\_from\\_NASIS\\_db](#)

---

fetchNASISWebReport    *Extract component tables from NASIS Web Reports*

---

**Description**

Extract component tables from NASIS Web Reports

**Usage**

```
fetchNASISWebReport(  
  projectname,  
  rmHzErrors = FALSE,  
  fill = FALSE,  
  stringsAsFactors = default.stringsAsFactors()  
)  
  
get_component_from_NASISWebReport(  
  projectname,  
  stringsAsFactors = default.stringsAsFactors()  
)  
  
get_chorizon_from_NASISWebReport(  
  projectname,  
  fill = FALSE,  
  stringsAsFactors = default.stringsAsFactors()  
)  
  
get_legend_from_NASISWebReport(  
  mlraoffice,  
  areasymbol,  
  droplevels = TRUE,  
  stringsAsFactors = default.stringsAsFactors()  
)  
  
get_lmuaoverlap_from_NASISWebReport(  
  areasymbol,
```

```

    droplevels = TRUE,
    stringsAsFactors = default.stringsAsFactors()
  )

  get_mapunit_from_NASISWebReport(
    areasymbol,
    droplevels = TRUE,
    stringsAsFactors = default.stringsAsFactors()
  )

  get_projectmapunit_from_NASISWebReport(
    projectname,
    stringsAsFactors = default.stringsAsFactors()
  )

  get_projectmapunit2_from_NASISWebReport(
    mlrassoarea,
    fiscalyear,
    projectname,
    stringsAsFactors = default.stringsAsFactors()
  )

  get_project_from_NASISWebReport(mlrassoarea, fiscalyear)

  get_progress_from_NASISWebReport(mlrassoarea, fiscalyear, projecttypename)

  get_project_correlation_from_NASISWebReport(
    mlrassoarea,
    fiscalyear,
    projectname
  )

```

### Arguments

projectname	text string vector of project names to be inserted into a SQL WHERE clause (default: NA)
rmHzErrors	should pedons with horizonation errors be removed from the results? (default: FALSE)
fill	should rows with missing component ids be removed (default: FALSE)
stringsAsFactors	logical: should character vectors be converted to factors? This argument is passed to the <code>unicode()</code> function. It does not convert those vectors that have been set outside of <code>unicode()</code> (i.e. hard coded). The 'factory-fresh' default is TRUE, but this can be changed by setting <code>options(stringsAsFactors = FALSE)</code>
mlraoffice	text string value identifying the MLRA Regional Soil Survey Office group name inserted into a SQL WHERE clause (default: NA)
areasymbol	text string value identifying the area symbol (e.g. IN001 or IN%) inserted into a SQL WHERE clause (default: NA) NULL (default: TRUE)

droplevels	logical: indicating whether to drop unused levels in classifying factors. This is useful when a class has large number of unused classes, which can waste space in tables and figures.
mlrassoarea	text string value identifying the MLRA Soil Survey Office areasymbol symbol inserted into a SQL WHERE clause (default: NA)
fiscalyear	text string value identifying the fiscal year inserted into a SQL WHERE clause (default: NA)
projecttypename	text string value identifying the project type name inserted into a SQL WHERE clause (default: NA)

**Value**

A data.frame or list with the results.

**Author(s)**

Stephen Roecker

**Examples**

```

if (requireNamespace("curl") &
    curl::has_internet() &
    require("aqp") &
    require("ggplot2") &
    require("gridExtra"))
) {
  # query soil components by projectname
  test = fetchNASISWebReport(
    "EVAL - MLRA 111A - Ross silt loam, 0 to 2 percent slopes, frequently flooded",
    stringsAsFactors = TRUE)
  test = test$spc

  # profile plot
  plot(test)

  # convert the data for depth plot
  clay_slice = horizons(slice(test, 0:200 ~ claytotal_l + claytotal_r + claytotal_h))
  names(clay_slice) <- gsub("claytotal_", "", names(clay_slice))

  om_slice = horizons(slice(test, 0:200 ~ om_l + om_r + om_h))
  names(om_slice) = gsub("om_", "", names(om_slice))

  test2 = rbind(data.frame(clay_slice, var = "clay"),
                data.frame(om_slice, var = "om"))
}

h = merge(test2, site(test)[c("dmuid", "coiid", "compname", "compct_r")],

```

```

        by = "coiid",
        all.x = TRUE
    )

# depth plot of clay content by soil component
gg_comp <- function(x) {
  ggplot(x) +
    geom_line(aes(y = r, x = hzdept_r)) +
    geom_line(aes(y = r, x = hzdept_r)) +
    geom_ribbon(aes(ymin = l, ymax = h, x = hzdept_r), alpha = 0.2) +
    xlim(200, 0) +
    xlab("depth (cm)") +
    facet_grid(var ~ dmuiid + paste(compname, compct_r)) +
    coord_flip()
}
g1 <- gg_comp(subset(h, var == "clay"))
g2 <- gg_comp(subset(h, var == "om"))

grid.arrange(g1, g2)

# query cosoilmoist (e.g. water table data) by mukey
# NA depths are interpreted as (???) with impute=TRUE argument
x <- get_cosoilmoist_from_NASISWebReport(
  "EVAL - MLRA 111A - Ross silt loam, 0 to 2 percent slopes, frequently flooded",
  stringsAsFactors = TRUE)

ggplot(x, aes(x = as.integer(month), y = dept_r, lty = status)) +
  geom_rect(aes(xmin = as.integer(month), xmax = as.integer(month) + 1,
               ymin = 0, ymax = max(x$ddepb_r),
               fill = flodfreqcl)) +
  geom_line(cex = 1) +
  geom_point() +
  geom_ribbon(aes(ymin = dept_l, ymax = dept_h), alpha = 0.2) +
  ylim(max(x$ddepb_r), 0) +
  xlab("month") + ylab("depth (cm)") +
  scale_x_continuous(breaks = 1:12, labels = month.abb, name="Month") +
  facet_wrap(~ paste0(compname, '(', compct_r, ')')) +
  ggtitle(paste0(x$nationalmusym[1],
                ': Water Table Levels from Component Soil Moisture Month Data'))
}

```

## Description

This function fetches a variety of data associated with named soil series, extracted from the USDA-NRCS Official Series Description text files and detailed soil survey (SSURGO). These data are periodically updated and made available via SoilWeb.

## Usage

```
fetchOSD(soils, colorState = "moist", extended = FALSE)
```

## Arguments

<code>soils</code>	a character vector of named soil series; case-insensitive
<code>colorState</code>	color state for horizon soil color visualization: "moist" or "dry"
<code>extended</code>	if TRUE additional soil series summary data are returned, see details

## Details

- [overview of all soil series query functions](#)
- [competing soil series](#)
- [siblings](#)

The standard set of "site" and "horizon" data are returned as a `SoilProfileCollection` object (extended=FALSE. The "extended" suite of summary data can be requested by setting extended=TRUE. The resulting object will be a list with the following elements:)

**SPC** `SoilProfileCollection` containing standards "site" and "horizon" data

**competing** competing soil series from the SC database snapshot

**geog\_assoc\_soils** geographically associated soils, extracted from named section in the OSD

**geomcomp** empirical probabilities for geomorphic component, derived from the current SSURGO snapshot

**hillpos** empirical probabilities for hillslope position, derived from the current SSURGO snapshot

**mntnpos** empirical probabilities for mountain slope position, derived from the current SSURGO snapshot

**terrace** empirical probabilities for river terrace position, derived from the current SSURGO snapshot

**flats** empirical probabilities for flat landscapes, derived from the current SSURGO snapshot

**pmkind** empirical probabilities for parent material kind, derived from the current SSURGO snapshot

**pmorigin** empirical probabilities for parent material origin, derived from the current SSURGO snapshot

**mlra** empirical MLRA membership values, derived from the current SSURGO snapshot

**climate** experimental climate summaries from PRISM stack

**metadata** metadata associated with SoilWeb cached summaries

When using `extended=TRUE`, there are a couple of scenarios in which series morphology contained in SPC do not fully match records in the associated series summaries (e.g. competing).

1. **A query for soil series that exist entirely outside of CONUS (e.g. PALAU).** - Climate summaries are empty data.frames because these summaries are currently generated from PRISM. We are working on a solution.
2. **A query for data within CONUS, but OSD morphology missing due to parsing error (e.g. formatting, typos).** - Extended summaries are present but morphology missing from SPC. A warning is issued.
3. **A query for multiple soil series, with one more more listed as "inactive" (e.g. BREADSPRINGS).** - Extended summaries are present but morphology missing from SPC. A warning is issued.

These last two cases are problematic for analysis that makes use of morphology and extended data, such as outlined in this tutorial on [competing soil series](#).

### Value

a `SoilProfileCollection` object containing basic soil morphology and taxonomic information.

### Author(s)

D.E. Beaudette

### References

USDA-NRCS OSD search tools: [https://www.nrcs.usda.gov/wps/portal/nrcs/detailfull/soils/home/?cid=nrcs142p2\\_053587](https://www.nrcs.usda.gov/wps/portal/nrcs/detailfull/soils/home/?cid=nrcs142p2_053587)

### See Also

[OSDquery](#), [siblings](#)

### Examples

```
if(requireNamespace("curl") &
  curl::has_internet()) {

  # soils of interest
  s.list <- c('musick', 'cecil', 'drummer', 'amador', 'pentz',
            'reiff', 'san joaquin', 'montpellier', 'grangeville', 'pollasky', 'ramona')

  # fetch and convert data into an SPC
  s.moist <- fetchOSD(s.list, colorState='moist')
  s.dry <- fetchOSD(s.list, colorState='dry')

  # plot profiles
  # moist soil colors
  if(require("aqp")) {

    par(mar=c(0,0,0,0), mfrow=c(2,1))
    plot(s.moist, name='hzname',
```

```
        cex.names=0.85, axis.line.offset=-4)
plot(s.dry, name='hzname',
      cex.names=0.85, axis.line.offset=-4)

# extended mode: return a list with SPC + summary tables
x <- fetchOSD(s.list, extended = TRUE, colorState = 'dry')

par(mar=c(0,0,1,1))
plot(x$SPC)
str(x, 1)
}
}
```

---

fetchPedonPC

*Fetch commonly used site/horizon data from a PedonPC v.5 database.*

---

### Description

Fetch commonly used site/horizon data from a version 5.x PedonPC database, return as a SoilProfileCollection object.

### Usage

```
fetchPedonPC(dsn)
```

### Arguments

dsn                    The path to a PedonPC version 6.x database

### Details

This function currently works only on Windows.

### Value

a SoilProfileCollection class object

### Note

This function attempts to do most of the boilerplate work when extracting site/horizon data from a PedonPC or local NASIS database. Pedons that have errors in their horization are excluded from the returned object, however, their IDs are printed on the console. See [getHzErrorsPedonPC](#) for a simple approach to identifying pedons with problematic horization. Records from the 'taxhistory' table are selected based on 1) most recent record, or 2) record with the least amount of missing data.

### Author(s)

D. E. Beaudette and J. M. Skovlin



**See Also**

[get\\_hz\\_data\\_from\\_pedon\\_db](#)

---

 fetchRaCA

*Get Rapid Carbon Assessment (RaCA) data*


---

**Description**

Get Rapid Carbon Assessment (RaCA) data via state, geographic bounding-box, RaCA site ID, or series query from the SoilWeb API.

**Usage**

```
fetchRaCA(
  series = NULL,
  bbox = NULL,
  state = NULL,
  rcasiteid = NULL,
  get.vnir = FALSE
)
```

**Arguments**

series	a soil series name; case-insensitive
bbox	a bounding box in WGS84 geographic coordinates e.g. c(-120, 37, -122, 38), constrained to a 5-degree block
state	a two-letter US state abbreviation; case-insensitive
rcasiteid	a RaCA site id (e.g. 'C1609C01')
get.vnir	logical, should associated VNIR spectra be downloaded? (see details)

**Details**

The VNIR spectra associated with RaCA data are quite large [each gzip-compressed VNIR spectra record is about 6.6kb], so requests for these data are disabled by default. Note that VNIR spectra can only be queried by soil series or geographic BBOX.

**Value**

pedons: a SoilProfileCollection object containing site/pedon/horizon data  
 trees: a data.frame object containing tree DBH and height  
 veg: a data.frame object containing plant species  
 stock: a data.frame object containing carbon quantities (stocks) at standardized depths  
 sample: a data.frame object containing sample-level bulk density and soil organic carbon values  
 spectra: a numeric matrix containing VNIR reflectance spectra from 350–2500 nm

**Author(s)**

D.E. Beaudette, USDA-NRCS staff

**References**

[https://www.nrcs.usda.gov/wps/portal/nrcs/detail/soils/survey/?cid=nrcs142p2\\_054164](https://www.nrcs.usda.gov/wps/portal/nrcs/detail/soils/survey/?cid=nrcs142p2_054164)

**See Also**

[fetchOSD](#)

**Examples**

```
if(requireNamespace("curl") &
  curl::has_internet()) {

  if(require(aqp)) {

    # search by series name
    s <- fetchRaCA(series='auburn')

    # search by bounding-box
    # s <- fetchRaCA(bbox=c(-120, 37, -122, 38))

    # check structure
    str(s, 1)

    # extract pedons
    p <- s$pedons

    # how many pedons
    length(p)

    # plot
    par(mar=c(0,0,0,0))
    plot(p, name='hzn_desgn', max.depth=150)
  }
}
```

---

fetchSCAN

*Fetch SCAN Data*

---

**Description**

Query soil/climate data from USDA-NRCS SCAN Stations (experimental)

**Usage**

```
fetchSCAN(site.code, year, report = "SCAN", req = NULL)
```

**Arguments**

site.code	a vector of site codes
year	a vector of years
report	report name, single value only
req	list of SCAN request parameters, for backwards-compatibility only

**Details**

See [the fetchSCAN tutorial for details](#). These functions require the `httr` and `rvest` libraries.

**Value**

a `data.frame` object

**Note**

`SCAN_sensor_metadata()` is known to crash on 32bit R / libraries (Windows).

**Author(s)**

D.E. Beaudette

**References**

<https://www.wcc.nrcs.usda.gov/index.html>

**Examples**

```
if(requireNamespace("curl") &
  curl::has_internet()) {

  # get data: new interface
  x <- fetchSCAN(site.code=c(356, 2072), year=c(2015, 2016))
  str(x)

  # get sensor metadata
  m <- SCAN_sensor_metadata(site.code=c(356, 2072))

  # get site metadata
  m <- SCAN_site_metadata(site.code=c(356, 2072))
}
```

---

 fetchSDA
 

---



---

*Download and Flatten Data from Soil Data Access*


---

**Description**

Functions to download and flatten commonly used tables and from Soil Data Access, and create soil profile collection objects (SPC).

**Usage**

```
fetchSDA(
  WHERE = NULL,
  duplicates = FALSE,
  childs = TRUE,
  nullFragAsZero = TRUE,
  rmHzErrors = FALSE,
  droplevels = TRUE,
  stringsAsFactors = default.stringsAsFactors()
)
```

**Arguments**

WHERE	text string formatted as an SQL WHERE clause (default: FALSE)
duplicates	logical; if TRUE a record is returned for each unique mukey (may be many per nationalmusym)
childs	logical; if FALSE parent material and geomorphic child tables are not flattened and appended
nullFragAsZero	should fragment volumes of NULL be interpreted as 0? (default: TRUE), see details
rmHzErrors	should pedons with horizonation errors be removed from the results? (default: FALSE)
droplevels	logical: indicating whether to drop unused levels in classifying factors. This is useful when a class has large number of unused classes, which can waste space in tables and figures.
stringsAsFactors	logical: should character vectors be converted to factors? This argument is passed to the uncode() function. It does not convert those vectors that have set outside of uncode() (i.e. hard coded). The 'factory-fresh' default is TRUE, but this can be changed by setting options(stringsAsFactors = FALSE)

**Details**

These functions return data from Soil Data Access with the use of a simple text string that formatted as an SQL WHERE clause (e.g. WHERE = "areasybol = 'IN001' ". All functions are SQL queries that wrap around SDAquery() and format the data for analysis.

Beware SDA includes the data for both SSURGO and STATSGO2. The areasymbol for STATSGO2 is US. For just SSURGO, include WHERE = "areasymbol != 'US'".

If the duplicates argument is set to TRUE, duplicate components are returned. This is not necessary with data returned from NASIS, which has one unique national map unit. SDA has duplicate map national map units, one for each legend it exists in.

The value of nullFragmentsAreZero will have a significant impact on the rock fragment fractions returned by fetchSDA. Set nullFragmentsAreZero = FALSE in those cases where there are many data-gaps and NULL rock fragment values should be interpreted as NULLs. Set nullFragmentsAreZero = TRUE in those cases where NULL rock fragment values should be interpreted as 0.

### Value

A data.frame or SoilProfileCollection object.

### Author(s)

Stephen Roecker

### See Also

[SDA\\_query](#)

### Examples

```
if (requireNamespace("curl") &
    curl::has_internet() &
    require(aqp) &
    require("ggplot2") &
    require("gridExtra") &
    require("viridisLite")) {

  # query soil components by areasymbol and musym
  test = fetchSDA(WHERE = "areasymbol = 'IN005' AND musym = 'MnpB2'")

  # profile plot
  plot(test)

  # convert the data for depth plot
  clay_slice = horizons(slice(test, 0:200 ~ claytotal_l + claytotal_r + claytotal_h))
  names(clay_slice) <- gsub("claytotal_", "", names(clay_slice))

  om_slice = horizons(slice(test, 0:200 ~ om_l + om_r + om_h))
  names(om_slice) = gsub("om_", "", names(om_slice))
}
```

```

test2 = rbind(data.frame(clay_slice, var = "clay"),
              data.frame(om_slice, var = "om")
            )

h = merge(test2, site(test)[c("nationalmusym", "cokey", "compname", "compct_r")],
          by = "cokey",
          all.x = TRUE
        )

# depth plot of clay content by soil component
gg_comp <- function(x) {
  ggplot(x) +
    geom_line(aes(y = r, x = hzdept_r)) +
    geom_line(aes(y = r, x = hzdept_r)) +
    geom_ribbon(aes(ymin = l, ymax = h, x = hzdept_r), alpha = 0.2) +
    xlim(200, 0) +
    xlab("depth (cm)") +
    facet_grid(var ~ nationalmusym + paste(compname, compct_r)) +
    coord_flip()
}

g1 <- gg_comp(subset(h, var == "clay"))
g2 <- gg_comp(subset(h, var == "om"))

grid.arrange(g1, g2)

# query cosoilmoist (e.g. water table data) by mukey
x <- get_cosoilmoist_from_SDA(WHERE = "mukey = '1395352'")

ggplot(x, aes(x = as.integer(month), y = dept_r, lty = status)) +
  geom_rect(aes(xmin = as.integer(month), xmax = as.integer(month) + 1,
               ymin = 0, ymax = max(x$depb_r),
               fill = flodfreqcl)) +
  geom_line(cex = 1) +
  geom_point() +
  geom_ribbon(aes(ymin = dept_l, ymax = dept_h), alpha = 0.2) +
  ylim(max(x$depb_r), 0) +
  xlab("month") + ylab("depth (cm)") +
  scale_x_continuous(breaks = 1:12, labels = month.abb, name="Month") +
  facet_wrap(~ paste0(compname, ' (', compct_r, ')')) +
  ggtitle(paste0(x$nationalmusym[1],
                ': Water Table Levels from Component Soil Moisture Month Data'))

# query all Miami major components
s <- get_component_from_SDA(WHERE = "compname = 'Miami' \n
                             AND majcompflag = 'Yes' AND areasymbol != 'US'")

# landform vs 3-D morphometry
test <- {
  subset(s, ! is.na(landform) | ! is.na(geompos)) ->. ;
}

```

```

split(., .$drainagecl, drop = TRUE) ->. ;
lapply(., function(x) {
  test = data.frame()
  test = as.data.frame(table(x$landform, x$geompos))
  test$compname = x$compname[1]
  test$drainagecl = x$drainagecl[1]
  names(test)[1:2] <- c("landform", "geompos")
  return(test)
}) ->. ;
do.call("rbind", .) ->. ;
.[.$Freq > 0, ] ->. ;
within(., {
  landform = reorder(factor(landform), Freq, max)
  geompos = reorder(factor(geompos), Freq, max)
  geompos = factor(geompos, levels = rev(levels(geompos)))
}) ->. ;
}
test$Freq2 <- cut(test$Freq,
  breaks = c(0, 5, 10, 25, 50, 100, 150),
  labels = c("<5", "5-10", "10-25", "25-50", "50-100", "100-150")
)
ggplot(test, aes(x = geompos, y = landform, fill = Freq2)) +
  geom_tile(alpha = 0.5) + facet_wrap(~ paste0(compname, "\n", drainagecl)) +
  discrete_scale("colour", "viridis", function(n) viridisLite::viridis(n)) +
  theme(aspect.ratio = 1, axis.text.x = element_text(angle = 45, hjust = 1, vjust = 1)) +
  ggtitle("Landform vs 3-D Morphometry for Miami Major Components on SDA")
}

```

---

fetchSDA\_spatial

*Query Soil Data Access and Return Spatial Data*


---

## Description

This is a high-level "fetch" method to facilitate spatial queries to Soil Data Access (SDA) based on mapunit key (mukey) and national mapunit symbol (nationalmusym) for mupolygon (SSURGO) or gsmmupolygon (STATSGO) geometry OR legend key (lkey) and area symbols (areasybol) for sapolygon (Soil Survey Area; SSA) geometry).

A Soil Data Access spatial query is made returning geometry and key identifying information about the mapunit or area of interest. Additional columns from the mapunit or legend table can be included using `add.fields` argument.

This function automatically "chunks" the input vector (using `soilDB::makeChunks`) of mapunit identifiers to minimize the likelihood of exceeding the SDA data request size. The number of

chunks varies with the `chunk.size` setting and the length of your input vector. If you are working with many mapunits and/or large extents, you may need to decrease this number in order to have more chunks.

Querying regions with complex mapping may require smaller `chunk.size`. Numerically adjacent IDs in the input vector may share common qualities (say, all from same soil survey area or region) which could cause specific chunks to perform "poorly" (slow or error) no matter what the chunk size is. Shuffling the order of the inputs using `sample` may help to eliminate problems related to this, depending on how you obtained your set of MUKEY/nationalmusym to query. One could feasibly use `muacres` as a heuristic to adjust for total acreage within chunks.

## Usage

```
fetchSDA_spatial(
  x,
  by.col = "mukey",
  method = "feature",
  geom.src = "mupolygon",
  db = "SSURGO",
  add.fields = NULL,
  chunk.size = 10,
  verbose = TRUE
)
```

## Arguments

<code>x</code>	A vector of MUKEYs / national mapunit symbols (for mupolygon geometry); OR legend keys (LKEY) / area symbols (for sapolygon geometry)
<code>by.col</code>	Column name containing mapunit identifier "mukey", "nmusym", or "areasymbol" for geom.src sapolygon; default is inferred from <code>is.numeric(x) == TRUE</code> for mukey or lkey and (nationalmusym or areasymbol otherwise.
<code>method</code>	geometry result type: "feature" returns polygons, "bbox" returns the bounding box of each polygon, and "point" returns a single point within each polygon.
<code>geom.src</code>	Either mupolygon or sapolygon
<code>db</code>	Default: SSURGO. When geom.src is mupolygon, use STATSGO polygon geometry instead of SSURGO by setting <code>db = "STATSGO"</code>
<code>add.fields</code>	Column names from mapunit or legend table to add to result. Must specify parent table name as the prefix <code>mapunit</code> before column name e.g. <code>mapunit.muname</code> .
<code>chunk.size</code>	How many queries should spatial request be divided into? Necessary for large results. Default: 10
<code>verbose</code>	Print messages?

## Details

Note that STATSGO data are fetched using `CLIPAREASYMBOL = 'US'` to avoid duplicating state and national subsets of the geometry.



**Value**

A Spatial\*DataFrame corresponding to SDA spatial data for all symbols requested. Default result contains geometry with attribute table containing unique feature ID, symbol and area symbol plus additional fields in result specified with `add.fields`.

**Author(s)**

Andrew G. Brown

**Examples**

```
if(requireNamespace("curl") &
  curl::has_internet()) {

  # get spatial data for a single mukey
  single.mukey <- fetchSDA_spatial(x = "2924882")

  # demonstrate fetching full extent (multi-mukey) of national musym
  full.extent.nmusym <- fetchSDA_spatial(x = "2x815", by = "nmusym")

  # compare extent of nmusym to single mukey within it
  if(require(sp)) {
    plot(full.extent.nmusym, col = "RED",border=0)
    plot(single.mukey, add = TRUE, col = "BLUE", border=0)
  }

  # demo adding a field (`muname`) to attribute table of result
  head(fetchSDA_spatial(x = "2x815", by="nmusym", add.fields="muname"))
}
```

---

fetchSoilGrids

*Fetch SoilGrids 250m properties information from point locations*

---

**Description**

This function obtains SoilGrids properties information (250m raster resolution) given a data.frame containing site IDs, latitudes and longitudes.

**Usage**

```
fetchSoilGrids(locations, loc.names = c("id", "lat", "lon"))
```

**Arguments**

<code>locations</code>	A data.frame containing 3 columns referring to site ID, latitude and longitude.
<code>loc.names</code>	Optional: Column names referring to site ID, latitude and longitude. Default: <code>c("id", "lat", "lon")</code>

**Details**

The depth intervals returned are: "0-5cm", "5-15cm", "15-30cm", "30-60cm", "60-100cm", "100-200cm" and the properties returned are "bdod", "cec", "cfvo", "clay", "nitrogen", "phh2o", "sand", "silt", "soc" – each with 5th, 50th, 95th, mean and uncertainty values. Point data requests are made through `properties/query` endpoint of the SoilGrids v2.0 REST API: <https://rest.soilgrids.org/soilgrids/v2.0/docs>

**Value**

A `SoilProfileCollection`

**Author(s)**

Andrew G. Brown

**Examples**

```
if(requireNamespace("curl") &
  curl::has_internet()) {

  library(aqp)

  your.points <- data.frame(id = c("A", "B"),
                           lat = c(37.9, 38.1),
                           lon = c(-120.3, -121.5),
                           stringsAsFactors = FALSE)

  x <- fetchSoilGrids(your.points)

  plotSPC(x, name = NA, color = "socQ50")
}
```

---

fetchVegdata

*Load most common vegplot data from local NASIS database*

---

**Description**

Load most common vegplot data from local NASIS database

**Usage**

```
fetchVegdata(
  SS = TRUE,
  stringsAsFactors = default.stringsAsFactors(),
  dsn = NULL
)
```

**Arguments**

SS	fetch data from the currently loaded selected set in NASIS or from the entire local database (default: TRUE)
stringsAsFactors	logical: should character vectors be converted to factors? This argument is passed to the uncode() function. It does not convert those vectors that have been set outside of uncode() (i.e. hard coded).
dsn	Optional: path to local SQLite database containing NASIS table structure; default: NULL

**Value**

A named list containing: "vegplot", "vegplotlocation", "vegplotrhi", "vegplotspecies", "vegtransect", "vegtransplantsum", 'vegsiteindexsum', "vegsiteindexdet", and "vegplottext" tables

---

filter_geochem	<i>Filter KSSL Geochemical Table</i>
----------------	--------------------------------------

---

**Description**

A function to subset KSSL "geochem" / elemental analysis result table to obtain rows/columns based on: column name, preparation code, major / trace element method.

**Usage**

```
filter_geochem(
  geochem,
  columns = NULL,
  prep_code = NULL,
  major_element_method = NULL,
  trace_element_method = NULL
)
```

**Arguments**

geochem	geochemical data, as returned by fetchKSSL
columns	Column name(s) to include in result
prep_code	Character vector of prep code(s) to include in result.
major_element_method	Character vector of major element method(s) to include in result.
trace_element_method	Character vector of trace element method(s) to include in result.

**Value**

A data.frame, subsetted according to the constraints specified in arguments.

**Author(s)**

Andrew G. Brown.

---

format\_SQL\_in\_statement

*Format vector of values into a string suitable for an SQL IN statement.*

---

**Description**

Concatenate a vector to SQL IN-compatible syntax: `letters[1:3]` becomes `('a', 'b', 'c')`. Values in `x` are first passed through `unique()`.

**Usage**

```
format_SQL_in_statement(x)
```

**Arguments**

`x`                    A character vector.

**Value**

A character vector (unit length) containing concatenated group syntax for use in SQL IN, with unique value found in `x`.

**Note**

Only character output is supported.

**Examples**

```
library(aqp)

# get some mukeys
q <- "select top(2) mukey from mapunit;"
mukeys <- SDA_query(q)

# format for use in an SQL IN statement
mukey.inst <- format_SQL_in_statement(mukeys$mukey)
mukey.inst

# make a more specific query: for component+horizon data, just for those mukeys
q2 <- sprintf("SELECT * FROM mapunit
              INNER JOIN component ON mapunit.mukey = component.mukey
              INNER JOIN chorizon ON component.cokey = chorizon.cokey
              WHERE mapunit.mukey IN %s;", mukey.inst)
```

```

# do the query
res <- SDA_query(q2)

# build a SoilProfileCollection from horizon-level records
depths(res) <- cokey ~ hzdept_r + hzdepb_r

# normalize mapunit/component level attributes to site-level for plot
site(res) <- ~ muname + mukey + compname + compct_r + taxclname

# make a nice label
res$labelname <- sprintf("%s (%s%s)", res$compname, res$compct_r, "%")

# major components only
res <- subset(res, compct_r >= 85)

# inspect plot of result
par(mar=c(0,0,0,0))
groupedProfilePlot(res, groups = "mukey", color = "hzname", cex.names=0.8,
                    id.style = "side", label = "labelname")

```

---

getHzErrorsNASIS      *Check pedon horizon table for logic errors*

---

### Description

Check pedon horizon table for logic errors

### Usage

```
getHzErrorsNASIS(strict = TRUE, SS = TRUE, dsn = NULL)
```

### Arguments

strict	how strict should horizon boundaries be checked for consistency: TRUE=more   FALSE=less
SS	fetch data from the currently loaded selected set in NASIS or from the entire local database (default: TRUE)
dsn	Optional: path to local SQLite database containing NASIS table structure; de- fault: NULL

### Value

A data.frame containing problematic records with columns: 'peiid', 'pedon\_id', 'hzdept', 'hzdepb', 'hzname'

get\_colors\_from\_NASIS\_db

*Extract Soil Color Data from a local NASIS Database*

---

### **Description**

Get, format, mix, and return color data from a NASIS database.

### **Usage**

```
get_colors_from_NASIS_db(SS = TRUE, dsn = NULL)
```

### **Arguments**

SS	fetch data from Selected Set in NASIS or from the entire local database (default: TRUE)
dsn	Optional: path to local SQLite database containing NASIS table structure; default: NULL

### **Details**

This function currently works only on Windows.

### **Value**

A data.frame with the results.

### **Author(s)**

Jay M. Skovlin and Dylan E. Beaudette

### **See Also**

[simplifyColorData](#), [get\\_hz\\_data\\_from\\_NASIS\\_db](#), [get\\_site\\_data\\_from\\_NASIS\\_db](#)

---

get\_colors\_from\_pedon\_db

*Extract Soil Color Data from a PedonPC Database*

---

### **Description**

Get, format, mix, and return color data from a PedonPC database.

### **Usage**

```
get_colors_from_pedon_db(dsn)
```

**Arguments**

dsn                    The path to a 'pedon.mdb' database.

**Details**

This function currently works only on Windows.

**Value**

A data.frame with the results.

**Author(s)**

Dylan E. Beaudette and Jay M. Skovlin

**See Also**

[get\\_hz\\_data\\_from\\_pedon\\_db](#), [get\\_site\\_data\\_from\\_pedon\\_db](#)

---

get\_comonth\_from\_NASIS\_db

*Extract component month data from a local NASIS Database*

---

**Description**

Extract component month data from a local NASIS Database.

**Usage**

```
get_comonth_from_NASIS_db(
  SS = TRUE,
  fill = FALSE,
  stringsAsFactors = default.stringsAsFactors(),
  dsn = NULL
)
```

**Arguments**

SS                    get data from the currently loaded Selected Set in NASIS or from the entire local database (default: TRUE)

fill                    should missing "month" rows in the comonth table be filled with NA (FALSE)

stringsAsFactors       logical: should character vectors be converted to factors? This argument is passed to the uncode() function. It does not convert those vectors that have set outside of uncode() (i.e. hard coded). The 'factory-fresh' default is TRUE, but this can be changed by setting options(stringsAsFactors = FALSE)

dsn                    Optional: path to local SQLite database containing NASIS table structure; default: NULL

**Details**

This function currently works only on Windows.

**Value**

A list with the results.

**Author(s)**

Stephen Roecker

**See Also**

[fetchNASIS](#)

**Examples**

```
if(local_NASIS_defined()) {  
  # query text note data  
  cm <- try(get_comonth_from_NASIS_db())  
  
  # show structure of component month data  
  str(cm)  
}
```

---

get\_component\_data\_from\_NASIS\_db

*Extract component data from a local NASIS Database*

---

**Description**

This function currently works only on Windows.

**Usage**

```
get_component_data_from_NASIS_db(  
  SS = TRUE,  
  stringsAsFactors = default.stringsAsFactors(),  
  dsn = NULL  
)
```



**Arguments**

SS	fetch data from the currently loaded selected set in NASIS or from the entire local database (default: TRUE)
stringsAsFactors	logical: should character vectors be converted to factors? This argument is passed to the <code>unicode()</code> function. It does not convert those vectors that have set outside of <code>unicode()</code> (i.e. hard coded). The 'factory-fresh' default is TRUE, but this can be changed by setting <code>options(stringsAsFactors = FALSE)</code>
dsn	Optional: path to local SQLite database containing NASIS table structure; default: NULL

**Value**

A list with the results.

**Author(s)**

Dylan E. Beaudette, Stephen Roecker, and Jay M. Skovlin

**See Also**

[fetchNASIS](#)

**Examples**

```
if(local_NASIS_defined()) {  
  # query text note data  
  fc <- try(get_component_data_from_NASIS_db())  
  
  # show structure of component data returned  
  str(fc)  
}
```

---

get\_cosoilmoist\_from\_NASIS

*Read and Flatten the Component Soil Moisture Tables*

---

**Description**

Read and flatten the component soil moisture month tables from a local NASIS Database.

**Usage**

```
get_cosoilmoist_from_NASIS(  
  SS = TRUE,  
  impute = TRUE,  
  stringsAsFactors = default.stringsAsFactors(),  
  dsn = NULL  
)
```

**Arguments**

SS	fetch data from the currently loaded selected set in NASIS or from the entire local database (default: TRUE)
impute	replace missing (i.e. NULL) values with "Not_Populated" for categorical data, or the "RV" for numeric data or 201 cm if the "RV" is also NULL (default: TRUE)
stringsAsFactors	logical: should character vectors be converted to factors? This argument is passed to the <code>unicode()</code> function. It does not convert those vectors that have set outside of <code>unicode()</code> (i.e. hard coded). The 'factory-fresh' default is TRUE, but this can be changed by setting <code>options(stringsAsFactors = FALSE)</code>
dsn	Optional: path to local SQLite database containing NASIS table structure; default: NULL

**Details**

The component soil moisture tables within NASIS house monthly data on flooding, ponding, and soil moisture status. The soil moisture status is used to specify the water table depth for components (e.g. `status == "Moist"`).

**Value**

A `data.frame`.

**Note**

This function currently works only on Windows.

**Author(s)**

S.M. Roecker

**See Also**

[fetchNASIS](#), [get\\_cosoilmoist\\_from\\_NASISWebReport](#), [get\\_cosoilmoist\\_from\\_SDA](#), [get\\_comonth\\_from\\_SDA](#)

**Examples**

```

if(local_NASIS_defined()) {
  # load cosoilmoist (e.g. water table data)
  test <- try(get_cosoilmoist_from_NASIS())

  # inspect
  if(!inherits(test, 'try-error')) {
    head(test)
  }
}

```

---

```
get_EDIT_ecoclass_by_geoUnit
```

*Get data.frame of all ecoclass for multiple EDIT geoUnit in ESD or ESG catalog*

---

**Description**

Supply a vector of target geoUnit. Data are accessed via Ecological Dynamics Interpretive Tool (EDIT) web services: <https://edit.jornada.nmsu.edu/resources/esd>

**Usage**

```
get_EDIT_ecoclass_by_geoUnit(geoUnit, catalog = c("esd", "esg"))
```

**Arguments**

geoUnit	A character vector of geoUnit codes e.g. c("018X", "022A") for MLRAs 18 and 22A.
catalog	Catalog ID. One of: "esd" or "esg"

**Value**

A data.frame containing: geoUnit, id, legacyId, name

**Examples**

```

if(requireNamespace("curl") &
  curl::has_internet()) {
  get_EDIT_ecoclass_by_geoUnit(c("018X", "022A"))
}

```

---

`get_extended_data_from_NASIS_db`*Extract accessory tables and summaries from a local NASIS Database*

---

**Description**

Extract accessory tables and summaries from a local NASIS Database

**Usage**

```
get_extended_data_from_NASIS_db(  
  SS = TRUE,  
  nullFragmentsAreZero = TRUE,  
  stringsAsFactors = default.stringsAsFactors(),  
  dsn = NULL  
)
```

**Arguments**

SS	get data from the currently loaded Selected Set in NASIS or from the entire local database (default: TRUE)
nullFragmentsAreZero	should fragment volumes of NULL be interpreted as 0? (default: TRUE), see details
stringsAsFactors	logical: should character vectors be converted to factors? This argument is passed to the <code>unicode()</code> function. It does not convert those vectors that have been set outside of <code>unicode()</code> (i.e. hard coded).
dsn	Optional: path to local SQLite database containing NASIS table structure; default: NULL

**Value**

A list with the results.

**Author(s)**

Jay M. Skovlin and Dylan E. Beaudette

**See Also**

[get\\_hz\\_data\\_from\\_NASIS\\_db](#), [get\\_site\\_data\\_from\\_NASIS\\_db](#)

## Examples

```
if(local_NASIS_defined()) {  
  # query extended data  
  e <- try(get_extended_data_from_NASIS_db())  
  
  # show contents of extended data  
  str(e)  
}
```

---

`get_extended_data_from_pedon_db`

*Extract accessory tables and summaries from a local pedonPC Database*

---

## Description

Extract accessory tables and summaries from a local pedonPC Database.

## Usage

```
get_extended_data_from_pedon_db(dsn)
```

## Arguments

`dsn`                    The path to a 'pedon.mdb' database.

## Details

This function currently works only on Windows.

## Value

A list with the results.

## Author(s)

Jay M. Skovlin and Dylan E. Beaudette

## See Also

[get\\_hz\\_data\\_from\\_pedon\\_db](#), [get\\_site\\_data\\_from\\_pedon\\_db](#)

---

`get_hz_data_from_NASIS_db`*Extract Horizon Data from a local NASIS Database*

---

**Description**

Get horizon-level data from a local NASIS database.

**Usage**

```
get_hz_data_from_NASIS_db(  
  SS = TRUE,  
  stringsAsFactors = default.stringsAsFactors(),  
  dsn = NULL  
)
```

**Arguments**

SS	fetch data from Selected Set in NASIS or from the entire local database (default: TRUE)
stringsAsFactors	logical: should character vectors be converted to factors? This argument is passed to the <code>unicode()</code> function. It does not convert those vectors that have been set outside of <code>unicode()</code> (i.e. hard coded).
dsn	Optional: path to local SQLite database containing NASIS table structure; default: NULL

**Value**

A `data.frame`.

**Note**

NULL total rock fragment values are assumed to represent an *absence* of rock fragments, and set to 0.

**Author(s)**

Jay M. Skovlin and Dylan E. Beaudette

**See Also**

[get\\_hz\\_data\\_from\\_NASIS\\_db](#), [get\\_site\\_data\\_from\\_NASIS\\_db](#)

---

get\_hz\_data\_from\_pedon\_db

*Extract Horizon Data from a PedonPC Database*

---

## Description

Get horizon-level data from a PedonPC database.

## Usage

```
get_hz_data_from_pedon_db(dsn)
```

## Arguments

dsn                    The path to a 'pedon.mdb' database.

## Details

This function currently works only on Windows.

## Value

A data.frame.

## Note

NULL total rock fragment values are assumed to represent an *absence* of rock fragments, and set to 0.

## Author(s)

Dylan E. Beaudette and Jay M. Skovlin

## See Also

[get\\_colors\\_from\\_pedon\\_db](#), [get\\_site\\_data\\_from\\_pedon\\_db](#)

---

`get_lablayer_data_from_NASIS_db`*Extract lab pedon layer data from a local NASIS Database*

---

**Description**

Get lab pedon layer-level (horizon-level) data from a local NASIS database.

**Usage**

```
get_lablayer_data_from_NASIS_db(SS = TRUE, dsn = NULL)
```

**Arguments**

SS	fetch data from the currently loaded selected set in NASIS or from the entire local database (default: TRUE)
dsn	Optional: path to local SQLite database containing NASIS table structure; default: NULL

**Value**

A data.frame.

**Note**

This function queries KSSL laboratory site/horizon data from a local NASIS database from the lab layer data table.

**Author(s)**

Jay M. Skovlin and Dylan E. Beaudette

**See Also**

[get\\_labpedon\\_data\\_from\\_NASIS\\_db](#)



---

get\_labpedon\_data\_from\_NASIS\_db

*Extract lab pedon data from a local NASIS Database*

---

### **Description**

Get lab pedon-level data from a local NASIS database.

### **Usage**

```
get_labpedon_data_from_NASIS_db(SS = TRUE, dsn = NULL)
```

### **Arguments**

SS	fetch data from the currently loaded selected set in NASIS or from the entire local database (default: TRUE)
dsn	Optional: path to local SQLite database containing NASIS table structure; default: NULL

### **Details**

This function currently works only on Windows, and requires a 'nasis\_local' ODBC connection.

### **Value**

A data.frame.

### **Note**

This function queries KSSL laboratory site/horizon data from a local NASIS database from the lab pedon data table.

### **Author(s)**

Jay M. Skovlin and Dylan E. Beaudette

### **See Also**

[get\\_lablayer\\_data\\_from\\_NASIS\\_db](#)

---

get_NOAA_GHCND	<i>Get Global Historical Climatology Network Daily (GHCND) data from NOAA API for given datatype(s), station IDs and years.</i>
----------------	---------------------------------------------------------------------------------------------------------------------------------

---

### Description

Obtain daily climatic summary data for a set of station IDs, years, and datatypes.

Note that typically results from the NOAA API are limited to 1000 records. However, by "chunking" up data into individual station`yeardatatypeid` combinations, record results generally do not exceed 365 records for daily summaries.

In order to use this function, you must obtain an API token from this website: <https://www.ncdc.noaa.gov/cdo-web/token>

### Usage

```
get_NOAA_GHCND(stations, years, datatypeids, apitoken)
```

### Arguments

stations	Station ID (e.g. GHCND:USC00388786)
years	One or more years (e.g. 2017:2020)
datatypeids	One or more NOAA GHCND data type IDs (e.g. c("PRCP", "SNOW"))
apitoken	API key token for NOAA NCDC web services ( <a href="https://www.ncdc.noaa.gov/cdo-web/token">https://www.ncdc.noaa.gov/cdo-web/token</a> )

### Value

A data.frame containing the GHCND data requested (limit 1000 records)

### Examples

```
## in order to use this function, you must obtain an API token from this website:
## https://www.ncdc.noaa.gov/cdo-web/token

# get_NOAA_GHCND(c("GHCND:USC00388786", "GHCND:USC00388787"),
#               years = 2017:2020,
#               datatypeids = c("PRCP", "SNOW"),
#               apitoken = "yourtokenhere")
```



---

get_OSD_JSON	<i>Fetch Official Series Description Raw Data in JSON format</i>
--------------	------------------------------------------------------------------

---

### Description

Fetch Official Series Description Raw Data in JSON format

### Usage

```
get_OSD_JSON(series, base_url = NULL)
```

### Arguments

series	A character vector of Official Series names e.g. "Chewacla"
base_url	Optional: alternate JSON repository path. Default: NULL uses <a href="https://github.com/ncss-tech/SoilKnowledgeBase">https://github.com/ncss-tech/SoilKnowledgeBase</a> .

### Details

The default `base_url` is to "raw" JSON files stored in a GitHub repository that is regularly updated from the official source of Series Descriptions. Using format: `https://raw.githubusercontent.com/ncss-tech/SoilKnowledgeBase/main/inst/extdata/OSD/{LETTER}/{SERIES}.json`

### Value

A `data.frame` with 1 row per series, and 1 column per "section" in the OSD as defined in National Soil Survey Handbook

### Examples

```
if(requireNamespace("curl") &
  curl::has_internet()) {
  series <- c("Musick", "Hector", "Chewacla")
  get_OSD_JSON(series)
}
```

---

get\_SDA\_hydric            *Get map unit hydric soils information from Soil Data Access*

---

**Description**

Get map unit hydric soils information from Soil Data Access

**Usage**

```
get_SDA_hydric(areasymbols = NULL, mukeys = NULL)
```

**Arguments**

areasymbols     vector of soil survey area symbols  
mukeys           vector of map unit keys

**Value**

a data.frame

**Author(s)**

Jason Nemecek, Chad Ferguson, Andrew Brown

---

get\_SDA\_interpretation            *Get map unit interpretations from Soil Data Access by rule name*

---

**Description**

Get map unit interpretations from Soil Data Access by rule name

**Usage**

```
get_SDA_interpretation(  
  rulename,  
  method = c("Dominant Component", "Dominant Condition", "Weighted Average", "None"),  
  areasymbols = NULL,  
  mukeys = NULL  
)
```

**Arguments**

rulename	rule name of interpretation (matching a mrulename in cointerp table)
method	aggregation method. One of: "Dominant Component", "Dominant Condition", "Weighted Average", "None". If "None" is selected one row will be returned per component, otherwise one row will be returned per map unit.
areasympols	vector of soil survey area symbols
mukeys	vector of map unit keys

**Details****Rule Names in cointerp table:**

- AGR-Agronomic Concerns (ND)
- AGR-Available Water Capacity (ND)
- AGR-Natural Fertility (ND)
- AGR-Pesticide and Nutrient Leaching Potential, NIRR (ND)
- AGR-Pesticide and Nutrient Runoff Potential (ND)
- AGR-Physical Limitations (ND)
- AGR-Rooting Depth (ND)
- AGR-Sodicity (ND)
- AGR-Subsurface Salinity (ND)
- AGR-Surface Crusting (ND)
- AGR-Surface Salinity (ND)
- AGR-Water Erosion (ND)
- AGR-Wind Erosion (ND)
- AGR - Air Quality; PM10 (TX)
- AGR - Air Quality; PM2\_5 (TX)
- AGR - Avocado Root Rot Hazard (CA)
- AGR - Barley Yield (MT)
- AGR - California Revised Storie Index (CA)
- AGR - Conventional Tillage (TX)
- AGR - Filter Strips (TX)
- AGR - Grape non-irrigated (MO)
- AGR - Hops Site Suitability (WA)
- AGR - Index for alfalfa hay, irrigated (NV)
- AGR - Map Unit Cropland Productivity (MN)
- AGR - Mulch Till (TX)
- AGR - Nitrate Leaching Potential, Irrigated (WA)
- AGR - Nitrate Leaching Potential, Nonirrigated (MA)
- AGR - Nitrate Leaching Potential, Nonirrigated (MT)
- AGR - Nitrate Leaching Potential, Nonirrigated (WA)
- AGR - No Till (TX)
- AGR - No Till (VT)

- AGR - No Till, Tile Drained (TX)
- AGR - Oats Yield (MT)
- AGR - Orchard Groups (TX)
- AGR - Pasture hayland (MO)
- AGR - Pesticide Loss Potential-Leaching
- AGR - Pesticide Loss Potential-Leaching (NE)
- AGR - Pesticide Loss Potential-Soil Surface Runoff
- AGR - Pesticide Loss Potential-Soil Surface Runoff (NE)
- AGR - Plant Growth Index PGI no Climate Adj. (TX)
- AGR - Plant Growth Index PGI with Climate Adj. (TX)
- AGR - Plant Growth Index PGI with Climate Adj. MAP,MAAT (TX)
- AGR - Rangeland Grass/Herbaceous Productivity Index (TX)
- AGR - Ridge Till (TX)
- AGR - Rutting Hazard =< 10,000 Pounds per Wheel (TX)
- AGR - Rutting Hazard > 10,000 Pounds per Wheel (TX)
- AGR - Selenium Leaching Potential (CO)
- AGR - Spring Wheat Yield (MT)
- AGR - Water Erosion Potential (TX)
- AGR - Water Erosion Potential Wide Ratings Array (TX)
- AGR - Wind Erosion Potential (TX)
- AGR - Wind Erosion Potential Wide Ratings Array (TX)
- AGR - Wine Grape Site Suitability (WA)
- AGR - Winter Wheat Yield (MT)
- Alaska Exempt Wetland Potential (AK)
- American Wine Grape Varieties Site Desirability (Long)
- American Wine Grape Varieties Site Desirability (Medium)
- American Wine Grape Varieties Site Desirability (Short)
- American Wine Grape Varieties Site Desirability (Very Long)
- AWM - Animal Mortality Disposal (Catastrophic) (MO)
- AWM - Filter Group (OH)
- AWM - Irrigation Disposal of Wastewater
- AWM - Irrigation Disposal of Wastewater (DE)
- AWM - Irrigation Disposal of Wastewater (MD)
- AWM - Irrigation Disposal of Wastewater (OH)
- AWM - Irrigation Disposal of Wastewater (VT)
- AWM - Land App of Municipal Sewage Sludge (DE)
- AWM - Land App of Municipal Sewage Sludge (MD)
- AWM - Land Application of Dry and Slurry Manure (TX)
- AWM - Land Application of Milk (CT)
- AWM - Land Application of Municipal Biosolids, spring (OR)
- AWM - Land Application of Municipal Biosolids, summer (OR)
- AWM - Land Application of Municipal Biosolids, winter (OR)

- AWM - Land Application of Municipal Sewage Sludge
- AWM - Land Application of Municipal Sewage Sludge (OH)
- AWM - Land Application of Municipal Sewage Sludge (VT)
- AWM - Large Animal Disposal, Pit (MN)
- AWM - Manure and Food Processing Waste
- AWM - Manure and Food Processing Waste (DE)
- AWM - Manure and Food Processing Waste (MD)
- AWM - Manure and Food Processing Waste (OH)
- AWM - Manure and Food Processing Waste (VT)
- AWM - Manure Stacking - Site Evaluation (TX)
- AWM - Overland Flow Process Treatment of Wastewater
- AWM - Overland Flow Process Treatment of Wastewater (VT)
- AWM - Phosphorus Management (TX)
- AWM - Rapid Infil Disposal of Wastewater (DE)
- AWM - Rapid Infil Disposal of Wastewater (MD)
- AWM - Rapid Infiltration Disposal of Wastewater
- AWM - Rapid Infiltration Disposal of Wastewater (VT)
- AWM - Sensitive Soil Features (MN)
- AWM - Slow Rate Process Treatment of Wastewater
- AWM - Slow Rate Process Treatment of Wastewater (VT)
- AWM - Vegetated Treatment Area (PIA)
- AWM - Waste Field Storage Area (VT)
- BLM-Reclamation Suitability (MT)
- BLM - Chaining Suitability
- BLM - Fencing
- BLM - Fire Damage Susceptibility
- BLM - Fugitive Dust Resistance
- BLM - Mechanical Treatment, Rolling Drum
- BLM - Mechanical Treatment, Shredder
- BLM - Medusahead Invasion Susceptibility
- BLM - Pygmy Rabbit Habitat Potential
- BLM - Rangeland Drill
- BLM - Rangeland Seeding, Colorado Plateau Ecoregion
- BLM - Rangeland Seeding, Great Basin Ecoregion
- BLM - Rangeland Tillage
- BLM - Site Degradation Susceptibility
- BLM - Soil Compaction Resistance
- BLM - Soil Restoration Potential
- BLM - Yellow Star-thistle Invasion Susceptibility
- CA Prime Farmland (CA)
- Capping Fill Gravity Septic System (DE)
- CLASS RULE - Depth to any bedrock kind (5 classes) (NPS)



- CLASS RULE - Depth to lithic bedrock (5 classes) (NPS)
- CLASS RULE - Depth to non-lithic bedrock (5 classes) (NPS)
- CLASS RULE - Depth to root limiting layer (5 classes) (NPS)
- CLASS RULE - Soil Inorganic Carbon kg/m<sup>2</sup> to 2m (NPS)
- CLASS RULE - Soil Organic Carbon kg/m<sup>2</sup> to 2m (NPS)
- CLR-cropland limitation for corn and soybeans (IN)
- CLR-pastureland limitation (IN)
- Commodity Crop Productivity Index (Corn) (WI)
- CPI - Alfalfa Hay, IRR - Eastern Idaho Plateaus (ID)
- CPI - Alfalfa Hay, IRR - Klamath Valley and Basins (OR)
- CPI - Alfalfa Hay, IRR - Snake River Plains (ID)
- CPI - Alfalfa Hay, NIRR - Eastern Idaho Plateaus (ID)
- CPI - Alfalfa Hay, NIRR - Palouse, Northern Rocky Mtns. (ID)
- CPI - Alfalfa Hay, NIRR - Palouse, Northern Rocky Mtns. (WA)
- CPI - Barley, IRR - Eastern Idaho Plateaus (ID)
- CPI - Barley, NIRR - Eastern Idaho Plateaus (ID)
- CPI - Grass Hay, IRR - Eastern Idaho Plateaus (ID)
- CPI - Grass Hay, IRR - Klamath Valleys and Basins (OR)
- CPI - Grass Hay, NIRR - Klamath Valleys and Basins (OR)
- CPI - Grass Hay, NIRR - Palouse, Northern Rocky Mtns. (ID)
- CPI - Grass Hay, NIRR - Palouse, Northern Rocky Mtns. (WA)
- CPI - Potatoes, IRR - Eastern Idaho Plateaus (ID)
- CPI - Potatoes, IRR - Snake River Plains (ID)
- CPI - Small Grains Productivity Index (AK)
- CPI - Small Grains, IRR - Snake River Plains (ID)
- CPI - Small Grains, NIRR - Palouse Prairies (ID)
- CPI - Small Grains, NIRR - Palouse Prairies (OR)
- CPI - Small Grains, NIRR - Palouse Prairies (WA)
- CPI - Small Grains, NIRR - Snake River Plains (ID)
- CPI - Wheat, IRR - Eastern Idaho Plateaus (ID)
- CPI - Wheat, NIRR - Eastern Idaho Plateaus (ID)
- CPI - Wild Hay, NIRR - Eastern Idaho Plateaus (ID)
- CPI - Wild Hay, NIRR - Palouse, Northern Rocky Mtns. (ID)
- CPI - Wild Hay, NIRR - Palouse, Northern Rocky Mtns. (WA)
- Deep Infiltration Systems
- DHS - Catastrophic Event, Large Animal Mortality, Burial
- DHS - Catastrophic Event, Large Animal Mortality, Incinerate
- DHS - Catastrophic Mortality, Large Animal Disposal, Pit
- DHS - Catastrophic Mortality, Large Animal Disposal, Trench
- DHS - Emergency Animal Mortality Disposal by Shallow Burial
- DHS - Emergency Land Disposal of Milk
- DHS - Potential for Radioactive Bioaccumulation

- DHS - Potential for Radioactive Sequestration
- DHS - Rubble and Debris Disposal, Large-Scale Event
- DHS - Site for Composting Facility - Subsurface
- DHS - Site for Composting Facility - Surface
- DHS - Suitability for Clay Liner Material
- DHS - Suitability for Composting Medium and Final Cover
- Elevated Sand Mound Septic System (DE)
- ENG - Animal Disposal by Composting (Catastrophic) (WV)
- ENG - Application of Municipal Sludge (TX)
- ENG - Aquifer Assessment - 7081 (MN)
- ENG - Closed-Loop Horizontal Geothermal Heat Pump (CT)
- ENG - Cohesive Soil Liner (MN)
- ENG - Construction Materials - Gravel Source (MN)
- ENG - Construction Materials - Sand Source (MN)
- ENG - Construction Materials; Gravel Source
- ENG - Construction Materials; Gravel Source (AK)
- ENG - Construction Materials; Gravel Source (CT)
- ENG - Construction Materials; Gravel Source (ID)
- ENG - Construction Materials; Gravel Source (IN)
- ENG - Construction Materials; Gravel Source (MI)
- ENG - Construction Materials; Gravel Source (NE)
- ENG - Construction Materials; Gravel Source (NY)
- ENG - Construction Materials; Gravel Source (OH)
- ENG - Construction Materials; Gravel Source (OR)
- ENG - Construction Materials; Gravel Source (VT)
- ENG - Construction Materials; Gravel Source (WA)
- ENG - Construction Materials; Reclamation
- ENG - Construction Materials; Reclamation (DE)
- ENG - Construction Materials; Reclamation (MD)
- ENG - Construction Materials; Reclamation (MI)
- ENG - Construction Materials; Reclamation (OH)
- ENG - Construction Materials; Roadfill
- ENG - Construction Materials; Roadfill (AK)
- ENG - Construction Materials; Roadfill (GA)
- ENG - Construction Materials; Roadfill (OH)
- ENG - Construction Materials; Sand Source
- ENG - Construction Materials; Sand Source (AK)
- ENG - Construction Materials; Sand Source (CT)
- ENG - Construction Materials; Sand Source (GA)
- ENG - Construction Materials; Sand Source (ID)
- ENG - Construction Materials; Sand Source (IN)
- ENG - Construction Materials; Sand Source (NY)

- ENG - Construction Materials; Sand Source (OH)
- ENG - Construction Materials; Sand Source (OR)
- ENG - Construction Materials; Sand Source (VT)
- ENG - Construction Materials; Sand Source (WA)
- ENG - Construction Materials; Topsoil
- ENG - Construction Materials; Topsoil (AK)
- ENG - Construction Materials; Topsoil (DE)
- ENG - Construction Materials; Topsoil (GA)
- ENG - Construction Materials; Topsoil (ID)
- ENG - Construction Materials; Topsoil (MD)
- ENG - Construction Materials; Topsoil (MI)
- ENG - Construction Materials; Topsoil (OH)
- ENG - Construction Materials; Topsoil (OR)
- ENG - Construction Materials; Topsoil (WA)
- ENG - Daily Cover for Landfill
- ENG - Daily Cover for Landfill (AK)
- ENG - Daily Cover for Landfill (OH)
- ENG - Disposal Field (NJ)
- ENG - Disposal Field Gravity (DE)
- ENG - Disposal Field Suitability Class (NJ)
- ENG - Disposal Field Type Inst (NJ)
- ENG - Dwellings W/O Basements
- ENG - Dwellings W/O Basements (OH)
- ENG - Dwellings With Basements
- ENG - Dwellings with Basements (AK)
- ENG - Dwellings With Basements (OH)
- ENG - Dwellings without Basements (AK)
- ENG - Large Animal Disposal, Pit (CT)
- ENG - Large Animal Disposal, Trench (CT)
- ENG - Lawn and Landscape (OH)
- ENG - Lawn, Landscape, Golf Fairway
- ENG - Lawn, landscape, golf fairway (CT)
- ENG - Lawn, Landscape, Golf Fairway (MI)
- ENG - Lawn, Landscape, Golf Fairway (VT)
- ENG - Local Roads and Streets
- ENG - Local Roads and Streets (AK)
- ENG - Local Roads and Streets (GA)
- ENG - Local Roads and Streets (OH)
- ENG - New Ohio Septic Rating (OH)
- ENG - On-Site Waste Water Absorption Fields (MO)
- ENG - On-Site Waste Water Lagoons (MO)
- ENG - OSHA Soil Types (TX)

- ENG - Pier Beam Building Foundations (TX)
- ENG - Sanitary Landfill (Area)
- ENG - Sanitary Landfill (Area) (AK)
- ENG - Sanitary Landfill (Area) (OH)
- ENG - Sanitary Landfill (Trench)
- ENG - Sanitary Landfill (Trench) (AK)
- ENG - Sanitary Landfill (Trench) (OH)
- ENG - Septage Application - Incorporation or Injection (MN)
- ENG - Septage Application - Surface (MN)
- ENG - Septic System; Disinfection, Surface Application (TX)
- ENG - Septic Tank Absorption Fields
- ENG - Septic Tank Absorption Fields - At-Grade (MN)
- ENG - Septic Tank Absorption Fields - Mound (MN)
- ENG - Septic Tank Absorption Fields - Trench (MN)
- ENG - Septic Tank Absorption Fields (AK)
- ENG - Septic Tank Absorption Fields (DE)
- ENG - Septic Tank Absorption Fields (FL)
- ENG - Septic Tank Absorption Fields (MD)
- ENG - Septic Tank Absorption Fields (NY)
- ENG - Septic Tank Absorption Fields (OH)
- ENG - Septic Tank Absorption Fields (TX)
- ENG - Septic Tank Leaching Chamber (TX)
- ENG - Septic Tank, Gravity Disposal (TX)
- ENG - Septic Tank, Subsurface Drip Irrigation (TX)
- ENG - Sewage Lagoons
- ENG - Sewage Lagoons (AK)
- ENG - Sewage Lagoons (OH)
- ENG - Shallow Excavations
- ENG - Shallow Excavations (AK)
- ENG - Shallow Excavations (MI)
- ENG - Shallow Excavations (OH)
- ENG - Small Commercial Buildings
- ENG - Small Commercial Buildings (OH)
- ENG - Soil Potential of Road Salt Applications (CT)
- ENG - Soil Potential Ratings of SSDS (CT)
- ENG - Source of Caliche (TX)
- ENG - Stormwater Management / Infiltration (NY)
- ENG - Stormwater Management / Ponds (NY)
- ENG - Stormwater Management / Wetlands (NY)
- ENG - Unpaved Local Roads and Streets
- Farm and Garden Composting Facility - Surface
- FOR-Biomass Harvest (WI)

- FOR-Construction Limitations for Haul Roads/Log Landings(ME)
- FOR - Biomass Harvest (MA)
- FOR - Black Walnut Suitability Index (KS)
- FOR - Black Walnut Suitability Index (MO)
- FOR - Compaction Potential (WA)
- FOR - Construction Limitations - Haul Roads/Log Landing (OH)
- FOR - Construction Limitations For Haul Roads (MI)
- FOR - Construction Limitations for Haul Roads/Log Landings
- FOR - Damage by Fire (OH)
- FOR - Displacement Hazard
- FOR - Displacement Potential (WA)
- FOR - General Harvest Season (ME)
- FOR - General Harvest Season (VT)
- FOR - Hand Planting Suitability
- FOR - Hand Planting Suitability (ME)
- FOR - Hand Planting Suitability, MO13 (DE)
- FOR - Hand Planting Suitability, MO13 (MD)
- FOR - Harvest Equipment Operability
- FOR - Harvest Equipment Operability (DE)
- FOR - Harvest Equipment Operability (MD)
- FOR - Harvest Equipment Operability (ME)
- FOR - Harvest Equipment Operability (MI)
- FOR - Harvest Equipment Operability (OH)
- FOR - Harvest Equipment Operability (VT)
- FOR - Log Landing Suitability
- FOR - Log Landing Suitability (ID)
- FOR - Log Landing Suitability (ME)
- FOR - Log Landing Suitability (MI)
- FOR - Log Landing Suitability (OR)
- FOR - Log Landing Suitability (VT)
- FOR - Log Landing Suitability (WA)
- FOR - Mechanical Planting Suitability
- FOR - Mechanical Planting Suitability (CT)
- FOR - Mechanical Planting Suitability (ME)
- FOR - Mechanical Planting Suitability (OH)
- FOR - Mechanical Planting Suitability, MO13 (DE)
- FOR - Mechanical Planting Suitability, MO13 (MD)
- FOR - Mechanical Site Preparation (Deep)
- FOR - Mechanical Site Preparation (Deep) (DE)
- FOR - Mechanical Site Preparation (Deep) (MD)
- FOR - Mechanical Site Preparation (Surface)
- FOR - Mechanical Site Preparation (Surface) (DE)

- FOR - Mechanical Site Preparation (Surface) (MD)
- FOR - Mechanical Site Preparation (Surface) (MI)
- FOR - Mechanical Site Preparation (Surface) (OH)
- FOR - Mechanical Site Preparation; Deep (CT)
- FOR - Mechanical Site Preparation; Surface (ME)
- FOR - Potential Erosion Hazard (Off-Road/Off-Trail)
- FOR - Potential Erosion Hazard (Off-Road/Off-Trail) (MI)
- FOR - Potential Erosion Hazard (Off-Road/Off-Trail) (OH)
- FOR - Potential Erosion Hazard (Road/Trail)
- FOR - Potential Erosion Hazard (Road/Trail) (PIA)
- FOR - Potential Erosion Hazard, Road/Trail, Spring Thaw (AK)
- FOR - Potential Fire Damage Hazard
- FOR - Potential Seedling Mortality
- FOR - Potential Seedling Mortality (FL)
- FOR - Potential Seedling Mortality (MI)
- FOR - Potential Seedling Mortality (OH)
- FOR - Potential Seedling Mortality (PIA)
- FOR - Potential Seedling Mortality (VT)
- FOR - Potential Seedling Mortality (ME)
- FOR - Potential Windthrow Hazard (ME)
- FOR - Potential Windthrow Hazard (MI)
- FOR - Potential Windthrow Hazard (NY)
- FOR - Potential Windthrow Hazard (VT)
- FOR - Puddling Hazard
- FOR - Puddling Potential (WA)
- FOR - Road Suitability (Natural Surface)
- FOR - Road Suitability (Natural Surface) (ID)
- FOR - Road Suitability (Natural Surface) (ME)
- FOR - Road Suitability (Natural Surface) (OH)
- FOR - Road Suitability (Natural Surface) (OR)
- FOR - Road Suitability (Natural Surface) (VT)
- FOR - Road Suitability (Natural Surface) (WA)
- FOR - Rutting Hazard by Month
- FOR - Rutting Hazard by Season
- FOR - Shortleaf pine littleleaf disease susceptibility
- FOR - Soil Compactibility Risk
- FOR - Soil Rutting Hazard
- FOR - Soil Rutting Hazard (ME)
- FOR - Soil Rutting Hazard (OH)
- FOR - Soil Sustainability Forest Biomass Harvesting (CT)
- FOR - White Oak Suitability (MO)
- FOR - Windthrow Hazard

- FOR - Windthrow Hazard (WA)
- FOR (USFS) - Road Construction/Maintenance (Natural Surface)
- FOTG - Indiana Corn Yield Calculation (IN)
- FOTG - Indiana Slippage Potential (IN)
- FOTG - Indiana Soy Bean Yield Calculation (IN)
- FOTG - Indiana Wheat Yield Calculation (IN)
- Fragile Soil Index
- Gravity Full Depth Septic System (DE)
- GRL-FSG-NP-W (MT)
- GRL - Excavations to 24 inches for Plastic Pipelines (TX)
- GRL - Fencing, 24 inch Post Depth (MT)
- GRL - Fencing, Post Depth =<24 inches
- GRL - Fencing, Post Depth =<36 inches
- GRL - Fencing, Post Depth Less Than 24 inches (TX)
- GRL - Fencing, Post Depth Less Than 36 inches (TX)
- GRL - Juniper Encroachment Potential (NM)
- GRL - NV range seeding (Wind C = 10) (NV)
- GRL - NV range seeding (Wind C = 100) (NV)
- GRL - NV range seeding (Wind C = 20) (NV)
- GRL - NV range seeding (Wind C = 30) (NV)
- GRL - NV range seeding (Wind C = 40) (NV)
- GRL - NV range seeding (Wind C = 50) (NV)
- GRL - NV range seeding (Wind C = 60) (NV)
- GRL - NV range seeding (Wind C = 80) (NV)
- GRL - NV range seeding (Wind C >= 160) (NV)
- GRL - Pasture and Hayland SG (OH)
- GRL - Ranch Access Roads (TX)
- GRL - Rangeland Chaining (TX)
- GRL - Rangeland Disking (TX)
- GRL - Rangeland Dozing/Grubbing (TX)
- GRL - Rangeland Planting by Mechanical Seeding (TX)
- GRL - Rangeland Prescribed Burning (TX)
- GRL - Rangeland Roller Chopping (TX)
- GRL - Rangeland Root Plowing (TX)
- GRL - Utah Juniper Encroachment Potential
- GRL - Western Juniper Encroachment Potential (OR)
- Ground-based Solar Arrays, Ballast Anchor Systems
- Ground-based Solar Arrays, Soil-penetrating Anchor Systems
- Ground Penetrating Radar Penetration
- Hybrid Wine Grape Varieties Site Desirability (Long)
- Hybrid Wine Grape Varieties Site Desirability (Medium)
- Hybrid Wine Grape Varieties Site Desirability (Short)

- Inland Wetlands (CT)
- IRR-restrictive features for irrigation (OH)
- ISDH Septic Tank Interpretation (IN)
- Land Application of Municipal Sewage Sludge (PA)
- Lined Retention Systems
- Low Pressure Pipe Septic System (DE)
- MIL - Bivouac Areas (DOD)
- MIL - Excavations Crew-Served Weapon Fighting Position (DOD)
- MIL - Excavations for Individual Fighting Position (DOD)
- MIL - Excavations for Vehicle Fighting Position (DOD)
- MIL - Helicopter Landing Zones (DOD)
- MIL - Trafficability Veh. Type 1 1-pass wet season (DOD)
- MIL - Trafficability Veh. Type 1 50-passes wet season (DOD)
- MIL - Trafficability Veh. Type 1 dry season (DOD)
- MIL - Trafficability Veh. Type 2 1-pass wet season (DOD)
- MIL - Trafficability Veh. Type 2 50-passes wet season (DOD)
- MIL - Trafficability Veh. Type 2 dry season (DOD)
- MIL - Trafficability Veh. Type 3 1-pass wet season (DOD)
- MIL - Trafficability Veh. Type 3 50-passes wet season (DOD)
- MIL - Trafficability Veh. Type 3 dry season (DOD)
- MIL - Trafficability Veh. Type 4 1-pass wet season (DOD)
- MIL - Trafficability Veh. Type 4 50-passes wet season (DOD)
- MIL - Trafficability Veh. Type 4 dry season (DOD)
- MIL - Trafficability Veh. Type 5 1-pass wet season (DOD)
- MIL - Trafficability Veh. Type 5 50-passes wet season (DOD)
- MIL - Trafficability Veh. Type 5 dry season (DOD)
- MIL - Trafficability Veh. Type 6 1-pass wet season (DOD)
- MIL - Trafficability Veh. Type 6 50-passes wet season (DOD)
- MIL - Trafficability Veh. Type 6 dry season (DOD)
- MIL - Trafficability Veh. Type 7 1-pass wet season (DOD)
- MIL - Trafficability Veh. Type 7 50-passes wet season (DOD)
- MIL - Trafficability Veh. Type 7 dry season (DOD)
- MT - Conservation Tree/Shrub Groups (MT)
- Muscadine Wine Grape Site Desirability (Very Long)
- NCCPI - Irrigated National Commodity Crop Productivity Index
- NCCPI - National Commodity Crop Productivity Index (Ver 3.0)
- NCCPI - NCCPI Corn Submodel (I)
- NCCPI - NCCPI Cotton Submodel (II)
- NCCPI - NCCPI Small Grains Submodel (II)
- NCCPI - NCCPI Soybeans Submodel (I)
- Nitrogen Loss Potential (ND)
- Permafrost Sensitivity (AK)



- Pressure Dose Capping Fill Septic System (DE)
- Pressure Dose Full Depth Septic System (DE)
- REC - Camp and Picnic Areas (AK)
- REC - Camp Areas (CT)
- REC - Camp Areas; Primitive (AK)
- REC - Foot and ATV Trails (AK)
- REC - Off-Road Motorcycle Trails (CT)
- REC - Paths and Trails (CT)
- REC - Picnic Areas (CT)
- REC - Playgrounds (AK)
- REC - Playgrounds (CT)
- RSK-risk assessment for manure application (OH)
- Salinity Risk Index, Discharge Model (ND)
- SAS - CMECS Substrate Class
- SAS - CMECS Substrate Origin
- SAS - CMECS Substrate Subclass
- SAS - CMECS Substrate Subclass/Group
- SAS - CMECS Substrate Subclass/Group/Subgroup
- SAS - Eastern Oyster Habitat Restoration Suitability
- SAS - Eelgrass Restoration Suitability
- SAS - Land Utilization of Dredged Materials
- SAS - Mooring Anchor - Deadweight
- SAS - Mooring Anchor - Mushroom
- SAS - Northern Quahog (Hard Clam) Habitat Suitability
- Septic System A/B Soil System (Alternate) (PA)
- Septic System At-Grade Bed (Alternate) (PA)
- Septic System At Grade Shallow Field (alternative) (WV)
- Septic System CO-OP RFS III w/At-Grade Bed (PA)
- Septic System CO-OP RFS III w/Drip Irrigation (PA)
- Septic System CO-OP RFS III w/Spray Irrigation (PA)
- Septic System Drip Irrigation (Alternate) (PA)
- Septic System Drip Irrigation (alternative) (WV)
- Septic System Dual Field Trench (conventional) (WV)
- Septic System Elevated Field (alternative) (WV)
- Septic System Free Access Sand Filter w/At-Grade Bed (PA)
- Septic System Free Access Sand Filter w/Drip Irrigation (PA)
- Septic System Free Access Sand Filterw/Spray Irrigation (PA)
- Septic System In Ground Bed (conventional) (PA)
- Septic System In Ground Trench (conventional) (PA)
- Septic System In Ground Trench (conventional) (WV)
- Septic System Low Pressure Pipe (alternative) (WV)
- Septic System Modified Subsurface Sand Filter (Alt.) (PA)

- Septic System Mound (alternative) (WV)
- Septic System Peat Based Option1 (UV & At-Grade Bed)Alt (PA)
- Septic System Peat Based Option1 w/At-Grade Bed (Alt.) (PA)
- Septic System Peat Based Option2 w/Spray Irrigation (PA)
- Septic System Peat Sys Opt3 w/Subsurface Sand Filter (PA)
- Septic System Sand Mound Bed or Trench (PA)
- Septic System Shallow In Ground Trench (conventional) (WV)
- Septic System Shallow Placement Pressure Dosed (Alt.) (PA)
- Septic System Spray Irrigation (PA)
- Septic System Steep Slope Mound (alternative) (WV)
- Septic System Steep Slope Sand Mound (Alternate) (PA)
- Septic System Subsurface Sand Filter Bed (conventional) (PA)
- Septic System Subsurface Sand Filter Trench (standard) (PA)
- Shallow Infiltration Systems
- SOH - Suitability for Aerobic Soil Organisms
- SOH - Agricultural Organic Soil Subsidence
- SOH - Concentration of Salts- Soil Surface
- SOH - Organic Matter Depletion
- SOH - Soil Surface Sealing
- SOH - Soil Susceptibility to Compaction
- Soil Habitat for Saprophyte Stage of Coccidioides
- SOIL HEALTH ASSESSMENT (NJ)
- Soil Vegetative Groups (CA)
- Surface Runoff Class (CA)
- Unlined Retention Systems
- URB - Commercial Brick Bldg; w/Reinforced Concrete Slab (TX)
- URB - Commercial Brick Buildings w/Concrete Slab (TX)
- URB - Commercial Metal Bldg; w/Concrete Slab (TX)
- URB - Commercial Metal Bldg; w/Reinforced Concrete Slab (TX)
- URB - Commercial Metal Buildings w/o Concrete Slab (TX)
- URB - Concrete Driveways and Sidewalks (TX)
- URB - Dwellings on Concrete Slab (TX)
- URB - Dwellings With Basements (TX)
- URB - Lawns and Ornamental Plantings (TX)
- URB - Reinforced Concrete Slab (TX)
- URB - Rural Residential Development on Concrete Slab (TX)
- URB - Rural Residential Development w/Basement (TX)
- URB - Urban Residential Development on Concrete Slab (TX)
- URB - Urban Residential Development w/Basement (TX)
- URB/REC - Camp Areas
- URB/REC - Camp Areas (GA)
- URB/REC - Camp Areas (HI)

- URB/REC - Camp Areas (MI)
- URB/REC - Camp Areas (OH)
- URB/REC - Golf Fairways (OH)
- URB/REC - Off-Road Motorcycle Trails
- URB/REC - Off-Road Motorcycle Trails (OH)
- URB/REC - Paths and Trails
- URB/REC - Paths and Trails (GA)
- URB/REC - Paths and Trails (MI)
- URB/REC - Paths and Trails (OH)
- URB/REC - Picnic Areas
- URB/REC - Picnic Areas (GA)
- URB/REC - Picnic Areas (MI)
- URB/REC - Picnic Areas (OH)
- URB/REC - Playgrounds
- URB/REC - Playgrounds (GA)
- URB/REC - Playgrounds (MI)
- URB/REC - Playgrounds (OH)
- Vinifera Wine Grape Site Desirability (Long to Medium)
- Vinifera Wine Grape Site Desirability (Long)
- Vinifera Wine Grape Site Desirability (Short to Medium)
- Vinifera Wine Grape Site Desirability (Short)
- WAQ - Soil Pesticide Absorbed Runoff Potential (TX)
- WAQ - Soil Pesticide Leaching Potential (TX)
- WAQ - Soil Pesticide Solution Runoff Potential (TX)
- WLF-Soil Suitability - Karner Blue Butterfly (WI)
- WLF - Burrowing Mammals & Reptiles (TX)
- WLF - Chufa for Turkey Forage (LA)
- WLF - Crawfish Aquaculture (TX)
- WLF - Desert Tortoise (CA)
- WLF - Desertic Herbaceous Plants (TX)
- WLF - Domestic Grasses & Legumes for Food and Cover (TX)
- WLF - Food Plots for Upland Wildlife < 2 Acres (TX)
- WLF - Freshwater Wetland Plants (TX)
- WLF - Gopher Tortoise Burrowing Suitability
- WLF - Grain & Seed Crops for Food and Cover (TX)
- WLF - Irr. Domestic Grasses & Legumes for Food & Cover (TX)
- WLF - Irrigated Freshwater Wetland Plants (TX)
- WLF - Irrigated Grain & Seed Crops for Food & Cover (TX)
- WLF - Irrigated Saline Water Wetland Plants (TX)
- WLF - Riparian Herbaceous Plants (TX)
- WLF - Riparian Shrubs, Vines, & Trees (TX)
- WLF - Saline Water Wetland Plants (TX)

- WLF - Upland Coniferous Trees (TX)
- WLF - Upland Deciduous Trees (TX)
- WLF - Upland Desertic Shrubs & Trees (TX)
- WLF - Upland Mixed Deciduous & Coniferous Trees (TX)
- WLF - Upland Native Herbaceous Plants (TX)
- WLF - Upland Shrubs & Vines (TX)
- WMS-Subsurface Water Management, Installation (ND)
- WMS-Subsurface Water Management, Outflow Quality (ND)
- WMS-Subsurface Water Management, Performance (ND)
- WMS - Constructing Grassed Waterways (OH)
- WMS - Constructing Grassed Waterways (TX)
- WMS - Constructing Terraces & Diversions (TX)
- WMS - Constructing Terraces and Diversions (OH)
- WMS - Drainage - (MI)
- WMS - Drainage (IL)
- WMS - Drainage (OH)
- WMS - Embankments, Dikes, and Levees
- WMS - Embankments, Dikes, and Levees (OH)
- WMS - Embankments, Dikes, and Levees (VT)
- WMS - Excavated Ponds (Aquifer-fed)
- WMS - Excavated Ponds (Aquifer-fed) (OH)
- WMS - Excavated Ponds (Aquifer-fed) (VT)
- WMS - Grape Production with Drip Irrigation (TX)
- WMS - Grassed Waterways - (MI)
- WMS - Irrigation, General
- WMS - Irrigation, Micro (above ground)
- WMS - Irrigation, Micro (above ground) (VT)
- WMS - Irrigation, Micro (subsurface drip)
- WMS - Irrigation, Micro (subsurface drip) (VT)
- WMS - Irrigation, Sprinkler (close spaced outlet drops)
- WMS - Irrigation, Sprinkler (general)
- WMS - Irrigation, Sprinkler (general) (VT)
- WMS - Irrigation, Surface (graded)
- WMS - Irrigation, Surface (level)
- WMS - Pond Reservoir Area
- WMS - Pond Reservoir Area (GA)
- WMS - Pond Reservoir Area (MI)
- WMS - Pond Reservoir Area (OH)
- WMS - Sprinkler Irrigation (MT)
- WMS - Sprinkler Irrigation RDC (IL)
- WMS - Subsurface Drains - Installation (VT)
- WMS - Subsurface Drains - Performance (VT)

- WMS - Subsurface Drains < 3 Feet Deep (TX)
- WMS - Subsurface Drains > 3 Feet Deep (TX)
- WMS - Subsurface Water Management, Outflow Quality
- WMS - Subsurface Water Management, System Installation
- WMS - Subsurface Water Management, System Performance
- WMS - Surface Drains (TX)
- WMS - Surface Irrigation Intake Family (TX)
- WMS - Surface Water Management, System

**Value**

a data.frame

**Author(s)**

Jason Nemecek, Chad Ferguson, Andrew Brown

---

get\_SDA\_muaggatt      *Get map unit aggregate attribute information from Soil Data Access*

---

**Description**

Get map unit aggregate attribute information from Soil Data Access

**Usage**

```
get_SDA_muaggatt(areasymbols = NULL, mukeys = NULL)
```

**Arguments**

areasymbols      vector of soil survey area symbols  
mukeys            vector of map unit keys

**Value**

a data.frame

**Author(s)**

Jason Nemecek, Chad Ferguson, Andrew Brown

---

get\_SDA\_pmgrouppname     *Get map unit parent material group information from Soil Data Access*

---

### Description

Uses "Dominant Component" aggregation method.

### Usage

```
get_SDA_pmgrouppname(areasymbols = NULL, mukeys = NULL, simplify = TRUE)
```

### Arguments

areasymbols	vector of soil survey area symbols
mukeys	vector of map unit keys
simplify	logical; group into generalized parent material groups? Default TRUE

### Value

a data.frame

### Author(s)

Jason Nemecek, Chad Ferguson, Andrew Brown

---

get\_SDA\_property     *Get map unit properties from Soil Data Access*

---

### Description

Get map unit properties from Soil Data Access

### Usage

```
get_SDA_property(
  property,
  method = c("Dominant Component (Category)", "Weighted Average", "Min/Max",
    "Dominant Component (Numeric)", "Dominant Condition", "None"),
  areasymbols = NULL,
  mukeys = NULL,
  top_depth = NULL,
  bottom_depth = NULL,
  FUN = NULL
)
```

**Arguments**

property	a label or column name from property dictionary
method	one of: "Dominant Component (Category)", "Weighted Average", "Min/Max", "Dominant Component (Numeric)", "Dominant Condition", or "None". If "None" is selected, the number of rows returned will depend on whether a component or horizon level property was selected, otherwise the result will be 1:1 with the number of map units.
areasympols	vector of soil survey area symbols
mukeys	vector of map unit keys
top_depth	Optional: a numeric value for upper boundary (top depth) used for method="weighted average" and "dominant component (numeric)"
bottom_depth	Optional: a numeric value for lower boundary (bottom depth) used for method="weighted average" and "dominant component (numeric)"
FUN	Optional: character representing SQL aggregation function either "MIN" or "MAX" for method="min/max"

**Details**

The property argument refers to one of the property names or columns specified in the tables below.

**Selected Component-level Properties:**

<b>Property (Component)</b>	<b>Column</b>
Range Production - Favorable Year	rsprod_h
Range Production - Normal Year	rsprod_r
Range Production - Unfavorable Year	rsprod_l
Corrosion of Steel	corsteel
Corrosion of Concrete	corcon
Drainage Class	drainagecl
Hydrologic Group	hydrp
Taxonomic Class Name	taxclname
Taxonomic Order	taxorder
Taxonomic Suborder	taxsuborder
Taxonomic Temperature Regime	taxtempregime
Wind Erodibility Group	weg
Wind Erodibility Index	wei
t Factor	tfact

**Selected Horizon-level Properties:**

<b>Property (Horizon)</b>	<b>Column</b>
0.1 bar H2O - Rep Value	wtenthbar_r
0.33 bar H2O - Rep Value	wthirdbar_r
15 bar H2O - Rep Value	wfifteenbar_r
Available Water Capacity - Rep Value	awc_r

Bray 1 Phosphate - Rep Value	pbray1_r
Bulk Density 0.1 bar H2O - Rep Value	dbtentibar_r
Bulk Density 0.33 bar H2O - Rep Value	dbthirdbar_r
Bulk Density 15 bar H2O - Rep Value	dbfifteenbar_r
Bulk Density oven dry - Rep Value	dbovendry_r
CaCO3 Clay - Rep Value	claysizedcarb_r
Calcium Carbonate - Rep Value	caco3_r
Cation Exchange Capacity - Rep Value	cec7_r
Coarse Sand - Rep Value	sandco_r
Coarse Silt - Rep Value	siltco_r
Effective Cation Exchange Capacity - Rep Value	ecec_r
Electrical Conductivity 1:5 by volume - Rep Value	ec15_r
Electrical Conductivity - Rep Value	ec_r
Exchangeable Sodium Percentage - Rep Value	esp_r
Extract Aluminum - Rep Value	extral_r
Extractable Acidity - Rep Value	extracid_r
Fine Sand - Rep Value	sandfine_r
Fine Silt - Rep Value	siltfine_r
Free Iron - Rep Value	freeiron_r
Gypsum - Rep Value	gypsum_r
Kf	kffact
Ki	kifact
Kr	krfact
Kw	kwfact
LEP - Rep Value	lep_r
Liquid Limit - Rep Value	ll_r
Medium Sand - Rep Value	sandmed_r
Organic Matter - Rep Value	om_r
Oxalate Aluminum - Rep Value	aloxalate_r
Oxalate Iron - Rep Value	feoxalate_r
Oxalate Phosphate - Rep Value	poxalate_r
Plasticity Index - Rep Value	pi_r
Rock Fragments 3 - 10 cm - Rep Value	frag3to10_r
Rock Fragments > 10 cm - Rep Value	fraggt10_r
Rubbed Fiber % - Rep Value	fiberrubbedpct_r
Satiated H2O - Rep Value	wsatiated_r
Saturated Hydraulic Conductivity - Rep Value	ksat_r
Sodium Adsorption Ratio - Rep Value	sar_r
Sum of Bases - Rep Value	sumbases_r
Total Clay - Rep Value	claytotal_r
Total Phosphate - Rep Value	ptotal_r
Total Rock Fragment Volume - Rep Value	fragvoltot
Total Sand - Rep Value	sandtotal_r
Total Silt - Rep Value	silttotal_r
Unrubbed Fiber % - Rep Value	fiberunrubbedpct_r
Very Coarse Sand - Rep Value	sandvc_r
Very Fine Sand - Rep Value	sandvf_r
Water Soluble Phosphate - Rep Value	ph2osoluble_r



no. 10 sieve - Rep Value	sieveno10_r
no. 200 sieve - Rep Value	sieveno200_r
no. 4 sieve - Rep Value	sieveno4_r
no. 40 sieve - Rep Value	sieveno40_r
pH .01M CaCl2 - Rep Value	ph01mcacl2_r
pH 1:1 water - Rep Value	ph1to1h2o_r
pH Oxidized - Rep Value	phoxidized_r

**Value**

a data.frame with result

**Author(s)**

Jason Nemecek, Chad Ferguson, Andrew Brown

---

get\_site\_data\_from\_NASIS\_db

*Extract Site Data from a local NASIS Database*

---

**Description**

Get site-level data from a local NASIS database.

**Usage**

```
get_site_data_from_NASIS_db(
  SS = TRUE,
  stringsAsFactors = default.stringsAsFactors(),
  dsn = NULL
)
```

**Arguments**

SS	fetch data from Selected Set in NASIS or from the entire local database (default: TRUE)
stringsAsFactors	logical: should character vectors be converted to factors? This argument is passed to the uncode() function. It does not convert those vectors that have been set outside of uncode() (i.e. hard coded).
dsn	Optional: path to local SQLite database containing NASIS table structure; default: NULL

**Details**

When multiple "site bedrock" entries are present, only the shallowest is returned by this function.

**Value**

A data.frame

**Author(s)**

Jay M. Skovlin and Dylan E. Beaudette

**See Also**

[get\\_hz\\_data\\_from\\_NASIS\\_db](#)

---

get\_site\_data\_from\_pedon\_db

*Extract Site Data from a PedonPC Database*

---

**Description**

Get site-level data from a PedonPC database.

**Usage**

```
get_site_data_from_pedon_db(dsn)
```

**Arguments**

dsn                    The path to a 'pedon.mdb' database.

**Value**

A data.frame.

**Note**

This function currently works only on Windows.

**Author(s)**

Dylan E. Beaudette and Jay M. Skovlin

**See Also**

[get\\_hz\\_data\\_from\\_pedon\\_db](#), [get\\_veg\\_from\\_AK\\_Site](#),

---

`get_soilseries_from_NASIS`*Get records from the Soil Classification (SC) database*

---

**Description**

These functions return records from the Soil Classification database, either from the local NASIS database (all series) or via web report (named series only).

**Usage**

```
get_soilseries_from_NASIS(  
  stringsAsFactors = default.stringsAsFactors(),  
  dsn = NULL  
)
```

**Arguments**`stringsAsFactors`

logical: should character vectors be converted to factors? This argument is passed to the `unicode()` function. It does not convert those vectors that have set outside of `unicode()` (i.e. hard coded).

`dsn`

Optional: path to local SQLite database containing NASIS table structure; default: NULL

**Value**

A `data.frame`

**Author(s)**

Stephen Roecker

---

`get_text_notes_from_NASIS_db`*Extract text note data from a local NASIS Database*

---

**Description**

Extract text note data from a local NASIS Database

**Usage**

```
get_text_notes_from_NASIS_db(SS = TRUE, fixLineEndings = TRUE, dsn = NULL)
```

**Arguments**

SS get data from the currently loaded Selected Set in NASIS or from the entire local database (default: TRUE)

fixLineEndings convert line endings from `\r\n` to `\n`

dsn Optional: path to local SQLite database containing NASIS table structure; default: NULL

**Value**

A list with the results.

**Author(s)**

Dylan E. Beaudette and Jay M. Skovlin

**See Also**

[get\\_hz\\_data\\_from\\_pedon\\_db](#), [get\\_site\\_data\\_from\\_pedon\\_db](#)

**Examples**

```
if(local_NASIS_defined()) {
  # query text note data
  t <- try(get_text_notes_from_NASIS_db())

  # show contents text note data, includes: siteobs, site, pedon, horizon level text notes data.
  str(t)

  # view text categories for site text notes
  if(!inherits(t, 'try-error')) {
    table(t$site_text$textcat)
  }
}
```

---

get\_veg\_data\_from\_NASIS\_db

*Extract veg data from a local NASIS Database*

---

**Description**

Extract veg data from a local NASIS Database.

**Usage**

```
get_veg_data_from_NASIS_db(SS = TRUE, dsn = NULL)
```

**Arguments**

SS	get data from the currently loaded Selected Set in NASIS or from the entire local database (default: TRUE)
dsn	Optional: path to local SQLite database containing NASIS table structure; default: NULL

**Details**

This function currently works only on Windows.

**Value**

A list with the results.

**Author(s)**

Jay M. Skovlin and Dylan E. Beaudette

**Examples**

```
if(local_NASIS_defined()) {  
  # query text note data  
  v <- try(get_veg_from_NASIS_db())  
  
  # show contents veg data returned  
  str(v)  
}
```

---

get\_veg\_from\_AK\_Site *Retrieve Vegetation Data from an AK Site Database*

---

**Description**

Retrieve Vegetation Data from an AK Site Database

**Usage**

```
get_veg_from_AK_Site(dsn)
```

**Arguments**

dsn	file path the the AK Site access database
-----	-------------------------------------------

**Value**

A data.frame with vegetation data in long format, linked to site ID.

**Note**

This function currently works only on Windows.

**Author(s)**

Dylan E. Beaudette

**See Also**

[get\\_hz\\_data\\_from\\_pedon\\_db](#), [get\\_site\\_data\\_from\\_pedon\\_db](#)

---

get\_veg\_from\_MT\_veg\_db

*Extract Site and Plot-level Data from a Montana RangeDB database*

---

**Description**

Get Site and Plot-level data from a Montana RangeDB database.

**Usage**

```
get_veg_from_MT_veg_db(dsn)
```

**Arguments**

dsn                    The name of the Montana RangeDB front-end database connection (see details).

**Details**

This function currently works only on Windows.

**Value**

A data.frame.

**Author(s)**

Jay M. Skovlin

**See Also**

[get\\_veg\\_species\\_from\\_MT\\_veg\\_db](#), [get\\_veg\\_other\\_from\\_MT\\_veg\\_db](#)

---

`get_veg_from_NPS_PLOTS_db`*Retrieve Vegetation Data from an NPS PLOTS Database*

---

**Description**

Used to extract species, stratum, and cover vegetation data from a backend NPS PLOTS Database. Currently works for any Microsoft Access database with an .mdb file format.

**Usage**`get_veg_from_NPS_PLOTS_db(dsn)`**Arguments**

dsn                    file path to the NPS PLOTS access database on your system.

**Value**

A data.frame with vegetation data in a long format with linkage to NRCS soil pedon data via the site\_id key field.

**Note**

This function currently only works on Windows.

**Author(s)**

Jay M. Skovlin

---

`get_veg_other_from_MT_veg_db`*Extract cover composition data from a Montana RangeDB database*

---

**Description**

Get cover composition data from a Montana RangeDB database.

**Usage**`get_veg_other_from_MT_veg_db(dsn)`**Arguments**

dsn                    The name of the Montana RangeDB front-end database connection (see details).

**Details**

This function currently works only on Windows.

**Value**

A data.frame.

**Author(s)**

Jay M. Skovlin

**See Also**

[get\\_veg\\_from\\_MT\\_veg\\_db](#), [get\\_veg\\_species\\_from\\_MT\\_veg\\_db](#)

---

`get_veg_species_from_MT_veg_db`

*Extract species-level Data from a Montana RangeDB database*

---

**Description**

Get species-level data from a Montana RangeDB database.

**Usage**

```
get_veg_species_from_MT_veg_db(dsn)
```

**Arguments**

`dsn`                    The name of the Montana RangeDB front-end database connection (see details).

**Details**

This function currently works only on Windows.

**Value**

A data.frame.

**Author(s)**

Jay M. Skovlin

**See Also**

[get\\_veg\\_from\\_MT\\_veg\\_db](#), [get\\_veg\\_other\\_from\\_MT\\_veg\\_db](#)



ISSR800.wcs

ISSR-800 Web Coverage Service (WCS)

**Description**

Intermediate-scale gridded (800m) soil property and interpretation maps from aggregated SSURGO and STATSGO data. These maps were developed by USDA-NRCS-SPSD staff in collaboration with UCD-LAWR. Originally for educational use and [interactive thematic maps](#), these data are a suitable alternative to gridded STATSGO-derived thematic soil maps. The full size grids can be [downloaded here](#).

**Usage**

```
ISSR800.wcs(aoi, var, res = 800, quiet = FALSE)
```

**Arguments**

aoi	area of interest (AOI) defined using a <code>Spatial*</code> , <code>RasterLayer</code> , <code>sf</code> , <code>sfc</code> or <code>bbox</code> object or a <code>list</code> , see details
var	ISSR-800 grid name, see details
res	grid resolution, units of meters. The native resolution of ISSR-800 grids (this WCS) is 800m.
quiet	logical, passed to <code>download.file</code> to enable / suppress URL and progress bar for download. #' @details aoi should be specified as a <code>Spatial*</code> , <code>RasterLayer</code> , <code>sf</code> , <code>sfc</code> , or <code>bbox</code> object or a <code>list</code> containing: aoi bounding-box specified as (xmin, ymin, xmax, ymax) e.g. <code>c(-114.16, 47.65, -114.08, 47.68)</code> crs coordinate reference system of BBOX, e.g. <code>'+init=epsg:4326'</code> The WCS query is parameterized using <code>raster::extent</code> derived from the above AOI specification, after conversion to the native CRS (EPSG:6350) of the ISSR-800 grids. Variables available from this WCS can be queried using <code>WCS_details(wcs = 'ISSR800')</code> .

**Value**

raster object containing indexed map unit keys and associated raster attribute table or a try-error if request fails

**Note**

There are still some issues to be resolved related to the encoding of NA Variables with a natural zero (e.g. SAR) have 0 set to NA.

**Author(s)**

D.E. Beaudette and A.G. Brown

KSSL\_VG\_model

*Develop a Water Retention Curve from KSSL Data***Description**

Water retention curve modeling via van Genuchten model and KSSL data.

**Usage**KSSL\_VG\_model(VG\_params, phi\_min = 10<sup>-6</sup>, phi\_max = 10<sup>8</sup>, pts = 100)**Arguments**

VG_params	data.frame or list object with the parameters of the van Genuchten model, see details
phi_min	lower limit for water potential in kPa
phi_max	upper limit for water potential in kPa
pts	number of points to include in estimated water retention curve

**Details**

This function was developed to work with measured or estimated parameters of the **van Genuchten model**, as generated by the **Rosetta model**. As such, VG\_params should have the following format and conventions:

**theta\_r** saturated water content, values should be in the range of {0, 1}

**theta\_s** residual water content, values should be in the range of {0, 1}

**alpha** related to the inverse of the air entry suction, function expects log10-transformed values with units of cm

**npar** index of pore size distribution, function expects log10-transformed values with units of 1/cm

**Value**

A list with the following components:

**VG\_curve** estimated water retention curve: paired estimates of water potential (phi) and water content (theta)

**VG\_function** spline function for converting water potential (phi, units of kPa) to estimated volumetric water content (theta, units of percent, range: {0, 1})

**VG\_inverse\_function** spline function for converting volumetric water content (theta, units of percent, range: {0, 1}) to estimated water potential (phi, units of kPa)

**Note**

A practical example is given in the [fetchSCAN tutorial](#).

**Author(s)**

D.E. Beaudette

**References**

[water retention curve estimation](#)

**Examples**

```
# basic example
d <- data.frame(
  theta_r = 0.0337216,
  theta_s = 0.4864061,
  alpha = -1.581517,
  npar = 0.1227247
)

vg <- KSSL_VG_model(d)

str(vg)
```

---

loafercreek

*Example SoilProfileCollection Objects Returned by fetchNASIS.*

---

**Description**

Several examples of soil profile collections returned by `fetchNASIS(from='pedons')` as `SoilProfileCollection` objects.

**Examples**

```
if(require("aqp")) {
# load example dataset
  data("gopheridge")

# what kind of object is this?
  class(gopheridge)

# how many profiles?
  length(gopheridge)
```

```

# there are 60 profiles, this calls for a split plot
par(mar=c(0,0,0,0), mfrow=c(2,1))

# plot soil colors
plot(gopheridge[1:30, ], name='hzname', color='soil_color')
plot(gopheridge[31:60, ], name='hzname', color='soil_color')

# need a larger top margin for legend
par(mar=c(0,0,4,0), mfrow=c(2,1))
# generate colors based on clay content
plot(gopheridge[1:30, ], name='hzname', color='clay')
plot(gopheridge[31:60, ], name='hzname', color='clay')

# single row and no labels
par(mar=c(0,0,0,0), mfrow=c(1,1))
# plot soils sorted by depth to contact
plot(gopheridge, name='', print.id=FALSE, plot.order=order(gopheridge$bedrckdepth))

# plot first 10 profiles
plot(gopheridge[1:10, ], name='hzname', color='soil_color', label='pedon_id', id.style='side')

# add rock fragment data to plot:
addVolumeFraction(gopheridge[1:10, ], colname='total_fragments_pct')

# add diagnostic horizons
addDiagnosticBracket(gopheridge[1:10, ], kind='argillic horizon', col='red', offset=-0.4)

## loafercreek
data("loafercreek")
# plot first 10 profiles
plot(loafercreek[1:10, ], name='hzname', color='soil_color', label='pedon_id', id.style='side')

# add rock fragment data to plot:
addVolumeFraction(loafercreek[1:10, ], colname='total_fragments_pct')

# add diagnostic horizons
addDiagnosticBracket(loafercreek[1:10, ], kind='argillic horizon', col='red', offset=-0.4)
}

```

---

local\_NASIS\_defined    *Check for presence of nasis\_local ODBC data source*

---

### Description

Check for presence of nasis\_local ODBC data source

### Usage

```
local_NASIS_defined(dsn = NULL)
```

**Arguments**

dsn                   Optional: path to local SQLite database containing NASIS table structure; default: NULL

**Value**

logical

**Examples**

```
if(local_NASIS_defined()) {  
  # use fetchNASIS or some other lower-level fetch function  
} else {  
  message('could not find `nasis_local` ODBC data source')  
}
```

---

makeChunks

*Generate chunk labels for splitting data*

---

**Description**

Generate chunk labels for splitting data

**Usage**

```
makeChunks(ids, size = 100)
```

**Arguments**

ids                   vector of IDs  
size                   chunk (group) size

**Value**

A numeric vector

**Examples**

```
# split the lowercase alphabet into 2 chunks  
  
aggregate(letters,  
  by = list(makeChunks(letters, size=13)),  
  FUN = paste0, collapse=",")
```

---

```
make_EDIT_service_URL  make_EDIT_service_URL
```

---

### Description

Construct a URL for Ecological Dynamics Interpretive Tool (EDIT) web services (<https://edit.jornada.nmsu.edu/services/...>) to return PDF, TXT or JSON results.

### Usage

```
make_EDIT_service_URL(
  src = c("descriptions", "downloads", "plant-community-tables", "models", "keys"),
  catalog = c("esd", "esg"),
  geoUnit = NULL,
  ecoclass = NULL,
  landuse = NULL,
  state = NULL,
  community = NULL,
  key = NULL,
  endpoint = NULL,
  querystring = NULL
)
```

### Arguments

src	One of: descriptions, downloads, plant-community-tables, models, keys
catalog	Catalog ID. One of: esd or esg
geoUnit	Geographic unit ID. For example: 022A
ecoclass	Ecological class ID. For example: F022AX101CA
landuse	Optional: Used only for src = "plant-community-tables"
state	Optional: Used only for src = "plant-community-tables"
community	Optional: Used only for src = "plant-community-tables"
key	Optional: Key number. All keys will be returned if not specified.
endpoint	Optional: Specific endpoint e.g. overview.json, class-list.json
querystring	Optional: Additional (optional) request parameters specified as a query string ?param1=value&param2=value. Used for class-list.json or PDF formatting parameters.

### Details

See the following official EDIT developer resources to see which endpoints are available for Ecological Site Description (ESD) or Ecological Site Group (ESG) catalogs:

- <https://edit.jornada.nmsu.edu/resources/esd>
- <https://edit.jornada.nmsu.edu/resources/esg>

**Value**

A character vector containing URLs with specified parameters. This function is vectorized.

**See Also**

get\_EDIT\_ecoclass\_by\_geoUnit

**Examples**

```
# url for all geoUnit keys as PDF
make_EDIT_service_URL(src = "descriptions",
                      catalog = "esd",
                      geoUnit = "039X")

# url for a single key within geoUnit as PDF
make_EDIT_service_URL(src = "descriptions",
                      catalog = "esd",
                      geoUnit = "039X",
                      key = "1")

# query for "full" description in JSON
desc <- make_EDIT_service_URL(src = "descriptions",
                             catalog = "esd",
                             geoUnit = "039X",
                             endpoint = "R039XA109AZ.json")

# query for "overview"
desc_ov <- make_EDIT_service_URL(src = "descriptions",
                                catalog = "esd",
                                geoUnit = "039X",
                                ecoclass = "R039XA109AZ",
                                endpoint = "overview.json")

# query for specific section, e.g. "water features"
desc_wf <- make_EDIT_service_URL(src = "descriptions",
                                catalog = "esd",
                                geoUnit = "039X",
                                ecoclass = "R039XA109AZ",
                                endpoint = "water-features.json")

# construct the URLs -- that is a query essentially
# then download the result with read_json

#full <- jsonlite::read_json(desc)
#overview <- jsonlite::read_json(desc_ov)
#waterfeature <- jsonlite::read_json(desc_wf)
```

---

 mapunit\_geom\_by\_ll\_bbox

*Fetch Map Unit Geometry from SDA*


---

## Description

Fetch map unit geometry from the SDA website by WGS84 bounding box. There is a limit on the amount of data returned as serialized JSON (~32Mb) and a total record limit of 100,000.

## Usage

```
mapunit_geom_by_ll_bbox(bbox, source = "sda")
```

## Arguments

bbox	a bounding box in WGS coordinates
source	the source database, currently limited to soil data access (SDA)

## Details

The SDA website can be found at <https://sdmdataaccess.nrcs.usda.gov>. See examples for bounding box formatting.

## Value

A SpatialPolygonsDataFrame of map unit polygons, in WGS84 (long,lat) coordinates.

## Note

SDA does not return the spatial intersection of map unit polygons and bounding box. Rather, just those polygons that are completely within the bounding box / overlap with the bbox. This function requires the 'rgdal' package.

## Author(s)

Dylan E. Beaudette

## Examples

```
## fetch map unit geometry from a bounding-box:
#
#           +----- (-120.41, 38.70)
#           |                               |
#           |                               |
# (-120.54, 38.61) -----+
#
if(requireNamespace("curl") &
```



```

curl::has_internet() &
require(sp) &
require(rgdal)) {

  # basic usage
  b <- c(-120.54,38.61,-120.41,38.70)
  x <- try(mapunit_geom_by_ll_bbox(b)) # about 20 seconds

  if(!inherits(x,'try-error')) {
    # note that the returned geometry is everything overlapping the bbox
    # and not an intersection... why?
    plot(x)
    rect(b[1], b[2], b[3], b[4], border='red', lwd=2)

    # get map unit data for matching map unit keys
    in.statement <- format_SQL_in_statement(unique(x$mukey))

    q <- paste("SELECT mukey, muname FROM mapunit WHERE mukey IN ", in.statement, sep="")
    res <- SDA_query(q)

    #inspect
    head(res)
  } else {
    message('could not download XML result from SDA')
  }
}

```

---

mix\_and\_clean\_colors    *Mix and Clean Colors*

---

## Description

Deprecated: only used in PedonPC functionality; use estimateColorMixture instead

## Usage

```
mix_and_clean_colors(x, wt = "pct", backTransform = FALSE)
```

## Arguments

x	a data.frame object containing sRGB coordinates ('r', 'g', 'b') in [0,1]
wt	fractional weights, usually area of hz face
backTransform	logical, should the mixed sRGB representation of soil color be transformed to closest Munsell chips? This is performed by

## Value

A data.frame containing mixed colors

---

 mukey.wcs

 gNATSGO / gSSURGO Map Unit Key Web Coverage Service (WCS)
 

---

### Description

Download chunks of the gNATSGO or gSSURGO map unit key grid via bounding-box from the SoilWeb WCS.

### Usage

```
mukey.wcs(aoi, db = c("gnatsgo", "gssurgo"), res = 30, quiet = FALSE)
```

### Arguments

aoi	area of interest (AOI) defined using either a Spatial*, RasterLayer, sf, sfc or bbox object, or a list, see details
db	name of the gridded map unit key grid to access, should be either 'gnatsgo' or 'gssurgo'
res	grid resolution, units of meters. The native resolution of gNATSGO and gSSURGO (this WCS) is 30m.
quiet	logical, passed to download.file to enable / suppress URL and progress bar for download.

### Details

aoi should be specified as one of: Spatial\*, RasterLayer, sf, sfc, bbox object, or a list containing:

aoi bounding-box specified as (xmin, ymin, xmax, ymax) e.g. c(-114.16, 47.65, -114.08, 47.68)

crs coordinate reference system of BBOX, e.g. '+init=epsg:4326'

The WCS query is parameterized using raster::extent derived from the above AOI specification, after conversion to the native CRS (EPSG:6350) of the gNATSGO / gSSURGO grid.

Databases available from this WCS can be queried using WCS\_details(wcs = 'mukey').

### Value

raster object containing indexed map unit keys and associated raster attribute table or a try-error if request fails

### Note

The gNATSGO grid includes raster soil survey map unit keys which are not in SDA.

### Author(s)

D.E. Beaudette and A.G. Brown

---

 NASIS\_table\_column\_keys

*NASIS 7 Tables, Columns and Foreign Keys*


---

### Description

This dataset contains NASIS 7 Tables, Columns and Foreign Keys

---

 OSDquery

*Full text searching of the USDA-NRCS Official Series Descriptions*


---

### Description

This is the R interface to [OSD search by Section](#) and [OSD Search](#) APIs provided by SoilWeb.

OSD records are searched with the [PostgreSQL fulltext indexing](#) and query system ([syntax details](#)). Each search field (except for the "brief narrative" and MLRA) corresponds with a section header in an OSD. The results may not include every OSD due to formatting errors and typos. Results are scored based on the number of times search terms match words in associated sections.

### Usage

```
OSDquery(
  everything = NULL,
  mlra = "",
  taxonomic_class = "",
  typical_pedon = "",
  brief_narrative = "",
  ric = "",
  use_and_veg = "",
  competing_series = "",
  geog_location = "",
  geog_assoc_soils = ""
)
```

### Arguments

everything	search entire OSD text (default is NULL), mlra may also be specified, all other arguments are ignored
mlra	a comma-delimited string of MLRA to search ('17,18,22A')
taxonomic_class	search family level classification
typical_pedon	search typical pedon section
brief_narrative	search brief narrative

<code>ric</code>	search range in characteristics section
<code>use_and_veg</code>	search use and vegetation section
<code>competing_series</code>	search competing series section
<code>geog_location</code>	search geographic setting section
<code>geog_assoc_soils</code>	search geographically associated soils section

### Details

See [this webpage](#) for more information.

- family level taxa are derived from SC database, not parsed OSD records
- MLRA are derived via spatial intersection (SSURGO x MLRA polygons)
- MLRA-filtering is only possible for series used in the current SSURGO snapshot (component name)
- logical AND: `&`
- logical OR: `|`
- wildcard, e.g. `rhy-something rhy:*`
- search terms with spaces need doubled single quotes: `"san joaquin"`
- combine search terms into a single expression: `(grano:* | granite)`

Related documentation can be found in the following tutorials

- [overview of all soil series query functions](#)
- [competing soil series](#)
- [siblings](#)

### Value

a `data.frame` object containing soil series names that match patterns supplied as arguments.

### Note

SoilWeb maintains a snapshot of the Official Series Description data.

### Author(s)

D.E. Beaudette

### References

[https://www.nrcs.usda.gov/wps/portal/nrcs/detailfull/soils/home/?cid=nrcs142p2\\_053587](https://www.nrcs.usda.gov/wps/portal/nrcs/detailfull/soils/home/?cid=nrcs142p2_053587)

### See Also

[fetchOSD](#), [siblings](#), [fetchOSD](#)

## Examples

```
if(requireNamespace("curl") &
    curl::has_internet() &
    require(aqp)) {

  # find all series that list Pardee as a geographically associated soil.
  s <- OSDquery(geog_assoc_soils = 'pardee')

  # get data for these series
  x <- fetchOSD(s$series, extended = TRUE, colorState = 'dry')

  # simple figure
  par(mar=c(0,0,1,1))
  plot(x$SPC)
}
```

---

parseWebReport

*Parse contents of a web report, based on supplied arguments.*

---

## Description

Parse contents of a web report, based on supplied arguments.

## Usage

```
parseWebReport(url, args, index = 1)
```

## Arguments

url	Base URL to a LIMS/NASIS web report.
args	List of named arguments to send to report, see details.
index	Integer index specifying the table to return, or, NULL for a list of tables

## Details

Report argument names can be inferred by inspection of the HTML source associated with any given web report.

## Value

A data.frame object in the case of a single integer passed to index, a list object in the case of an integer vector or NULL passed to index.

**Note**

Most web reports are for internal use only.

**Author(s)**

D.E. Beaudette and S.M. Roecker

**Examples**

```
# pending
```

---

processSDA_WKT	<i>Post-process WKT returned from SDA.</i>
----------------	--------------------------------------------

---

**Description**

This is a helper function, commonly used with SDA\_query to extract WKT (well-known text) representation of geometry to an sp-class object.

**Usage**

```
processSDA_WKT(d, g = "geom", p4s = "+proj=longlat +datum=WGS84")
```

**Arguments**

d	data.frame returned by SDA_query, containing WKT representation of geometry
g	name of column in d containing WKT geometry
p4s	PROJ4 CRS definition, typically GCS WGS84

**Details**

The SDA website can be found at <https://sdmdataaccess.nrcs.usda.gov>. See the [SDA Tutorial](#) for detailed examples.

**Value**

A Spatial\* object.

**Note**

This function requires the httr, jsonlite, XML, and rgeos packages.

**Author(s)**

D.E. Beaudette

ROSETTA

*ROSETTA Model API***Description**

A simple interface to the **ROSETTA model** for predicting hydraulic parameters from soil properties. The ROSETTA API was developed by Dr. Todd Skaggs (USDA-ARS) and links to the work of Zhang and Schaap, (2017). See the [related tutorial](#) for additional examples.

**Usage**

```
ROSETTA(x, vars, v = c("1", "2", "3"), chunkSize = 10000, conf = NULL)
```

**Arguments**

x	a data.frame of required soil properties, may contain other columns, see details
vars	character vector of column names in x containing relevant soil property values, see details
v	ROSETTA model version number: '1', '2', or '3', see details and references.
chunkSize	number of records per API call
conf	configuration passed to <code>httr::POST()</code> such as <code>verbose()</code> .

**Details**

Soil properties supplied in x must be described, in order, via vars argument. The API does not use the names but column ordering must follow: sand, silt, clay, bulk density, volumetric water content at 33kPa (1/3 bar), and volumetric water content at 1500kPa (15 bar).

The ROSETTA model relies on a minimum of 3 soil properties, with increasing (expected) accuracy as additional properties are included:

- required, sand, silt, clay: USDA soil texture separates (percentages) that sum to 100\
- optional, bulk density (any moisture basis): mass per volume after accounting for >2mm fragments, units of gm/cm<sup>3</sup>
- optional, volumetric water content at 33 kPa: roughly "field capacity" for most soils, units of cm<sup>3</sup>/cm<sup>3</sup>
- optional, volumetric water content at 1500 kPa: roughly "permanent wilting point" for most plants, units of cm<sup>3</sup>/cm<sup>3</sup>

Column names not specified in vars are retained in the output.

Three versions of the ROSETTA model are available, selected using v = 1, v = 2, or v = 3.

- version 1 - Schaap, M.G., F.J. Leij, and M.Th. van Genuchten. 2001. ROSETTA: a computer program for estimating soil hydraulic parameters with hierarchical pedotransfer functions. *Journal of Hydrology* 251(3-4): 163-176. doi: [10.1016/S00221694\(01\)004668](https://doi.org/10.1016/S00221694(01)004668).
- version 2 - Schaap, M.G., A. Nemes, and M.T. van Genuchten. 2004. Comparison of Models for Indirect Estimation of Water Retention and Available Water in Surface Soils. *Vadose Zone Journal* 3(4): 1455-1463. doi: [10.2136/vzj2004.1455](https://doi.org/10.2136/vzj2004.1455).
- version 3 - Zhang, Y., and M.G. Schaap. 2017. Weighted recalibration of the Rosetta pedotransfer model with improved estimates of hydraulic parameter distributions and summary statistics (Rosetta3). *Journal of Hydrology* 547: 39-53. doi: [10.1016/j.jhydrol.2017.01.004](https://doi.org/10.1016/j.jhydrol.2017.01.004).

### Author(s)

D.E. Beaudette, Todd Skaggs (ARS), Richard Reid

### References

- Consider using the interactive version, with copy/paste functionality at: <https://www.handbook60.org/rosetta>.
- Rosetta Model Home Page: <https://www.ars.usda.gov/pacific-west-area/riverside-ca/agricultural-water-efficiency-and-salinity-research-unit/docs/model/rosetta-model/>.
- Python ROSETTA model: <http://www.u.arizona.edu/~ygzhang/download.html>.
- Yonggen Zhang, Marcel G. Schaap. 2017. Weighted recalibration of the Rosetta pedotransfer model with improved estimates of hydraulic parameter distributions and summary statistics (Rosetta3). *Journal of Hydrology*. 547: 39-53. doi: [10.1016/j.jhydrol.2017.01.004](https://doi.org/10.1016/j.jhydrol.2017.01.004).
- Kosugi, K. 1999. General model for unsaturated hydraulic conductivity for soils with lognormal pore-size distribution. *Soil Sci. Soc. Am. J.* 63:270-277.
- Mualem, Y. 1976. A new model predicting the hydraulic conductivity of unsaturated porous media. *Water Resour. Res.* 12:513-522.
- Schaap, M.G. and W. Bouten. 1996. Modeling water retention curves of sandy soils using neural networks. *Water Resour. Res.* 32:3033-3040.
- Schaap, M.G., Leij F.J. and van Genuchten M.Th. 1998. Neural network analysis for hierarchical prediction of soil water retention and saturated hydraulic conductivity. *Soil Sci. Soc. Am. J.* 62:847-855.
- Schaap, M.G., and F.J. Leij, 1998. Database Related Accuracy and Uncertainty of Pedotransfer Functions, *Soil Science* 163:765-779.
- Schaap, M.G., F.J. Leij and M. Th. van Genuchten. 1999. A bootstrap-neural network approach to predict soil hydraulic parameters. In: van Genuchten, M.Th., F.J. Leij, and L. Wu (eds), *Proc. Int. Workshop, Characterization and Measurements of the Hydraulic Properties of Unsaturated Porous Media*, pp 1237-1250, University of California, Riverside, CA.
- Schaap, M.G., F.J. Leij, 1999, Improved prediction of unsaturated hydraulic conductivity with the Mualem-van Genuchten, Submitted to *Soil Sci. Soc. Am. J.*
- van Genuchten, M.Th. 1980. A closed-form equation for predicting the hydraulic conductivity of unsaturated soils. *Soil Sci. Am. J.* 44:892-898.



Schaap, M.G., F.J. Leij, and M.Th. van Genuchten. 2001. ROSETTA: a computer program for estimating soil hydraulic parameters with hierarchical pedotransfer functions. *Journal of Hydrology* 251(3-4): 163-176. doi: [10.1016/S00221694\(01\)004668](https://doi.org/10.1016/S00221694(01)004668).

Schaap, M.G., A. Nemes, and M.T. van Genuchten. 2004. Comparison of Models for Indirect Estimation of Water Retention and Available Water in Surface Soils. *Vadose Zone Journal* 3(4): 1455-1463. doi: [10.2136/vzj2004.1455](https://doi.org/10.2136/vzj2004.1455).

Zhang, Y., and M.G. Schaap. 2017. Weighted recalibration of the Rosetta pedotransfer model with improved estimates of hydraulic parameter distributions and summary statistics (Rosetta3). *Journal of Hydrology* 547: 39-53. doi: [10.1016/j.jhydrol.2017.01.004](https://doi.org/10.1016/j.jhydrol.2017.01.004).

---

SCAN\_SNOTEL\_metadata    *SCAN and SNOTEL Station Metadata*

---

## Description

SCAN and SNOTEL station metadata, a work in progress.

## Format

A data frame with 1092 observations on the following 12 variables.

**list("Name")** station name

**list("Site")** station ID

**list("State")** state

**list("Network")** sensor network: SCAN / SNOTEL

**list("County")** county

**list("Elevation\_ft")** station elevation in feet

**list("Latitude")** latitude of station

**list("Longitude")** longitude of station

**list("HUC")** associated watershed

**list("climstanm")** climate station name (TODO: remove this column)

**list("upedonid")** associated user pedon ID

**list("pedlabsampnum")** associated lab sample ID

## Details

These data have been compiled from several sources and represent a progressive effort to organize SCAN/SNOTEL station metadata. Therefore, some records may be missing or incorrect.

---

`SDA_query`*Soil Data Access Query*

---

**Description**

Submit a query to the Soil Data Access (SDA) REST/JSON web-service and return the results as a data.frame. There is a 100,000 record limit and 32Mb JSON serializer limit, per query. Queries should contain a WHERE statement or JOIN condition to limit the number of rows affected / returned. Consider wrapping calls to SDA\_query in a function that can iterate over logical chunks (e.g. areasymbol, mukey, cokey, etc.). The function makeChunks can help with such iteration.

**Usage**`SDA_query(q)`**Arguments**

`q` A valid T-SQL query surrounded by double quotes

**Details**

The SDA website can be found at <https://sdmdataaccess.nrcs.usda.gov> and query examples can be found at <https://sdmdataaccess.nrcs.usda.gov/QueryHelp.aspx>. A library of query examples can be found at [https://nasis.sc.egov.usda.gov/NasisReportsWebSite/lmsreport.aspx?report\\_name=SDA-SQL\\_Library\\_Home](https://nasis.sc.egov.usda.gov/NasisReportsWebSite/lmsreport.aspx?report_name=SDA-SQL_Library_Home).

SSURGO (detailed soil survey) and STATSGO (generalized soil survey) data are stored together within SDA. This means that queries that don't specify an area symbol may result in a mixture of SSURGO and STATSGO records. See the examples below and the [SDA Tutorial](#) for details.

**Value**

a data.frame result (NULL if empty, try-error on error)

**Note**

This function requires the `httr`, `jsonlite`, and `XML` packages

**Author(s)**

D.E. Beaudette

**See Also**

[mapunit\\_geom\\_by\\_ll\\_bbox](#)

**Examples**

```

if(requireNamespace("curl") &
  curl::has_internet()) {

  ## get SSURGO export date for all soil survey areas in California
  # there is no need to filter STATSGO
  # because we are filtering on SSURGO area symbols
  q <- "SELECT areasymbol, saverest FROM sacatalog WHERE areasymbol LIKE 'CA%';"
  x <- SDA_query(q)
  head(x)

  ## get SSURGO component data associated with the
  ## Amador series / major component only
  # this query must explicitly filter out STATSGO data
  q <- "SELECT cokey, compname, compcpt_r FROM legend
  INNER JOIN mapunit mu ON mu.lkey = legend.lkey
  INNER JOIN component co ON mu.mukey = co.mukey
  WHERE legend.areasymbol != 'US' AND compname = 'Amador';"

  res <- SDA_query(q)
  str(res)

  ## get component-level data for a specific soil survey area (Yolo county, CA)
  # there is no need to filter STATSGO because the query contains
  # an implicit selection of SSURGO data by areasymbol
  q <- "SELECT
  component.mukey, cokey, compcpt_r, compname, taxclname,
  taxorder, taxsuborder, taxgrtgroup, taxsubgrp
  FROM legend
  INNER JOIN mapunit ON mapunit.lkey = legend.lkey
  LEFT OUTER JOIN component ON component.mukey = mapunit.mukey
  WHERE legend.areasymbol = 'CA113' ;"

  res <- SDA_query(q)
  str(res)

  ## get tabular data based on result from spatial query
  # there is no need to filter STATSGO because
  # SDA_Get_Mukey_from_intersection_with_WktWgs84() implies SSURGO
  #
  # requires raster and rgeos packages because raster is suggested
  # and rgeos is additional
  if(require(raster) & require(rgeos)) {
    # text -> bbox -> WKT
    # xmin, xmax, ymin, ymax
    b <- c(-120.9, -120.8, 37.7, 37.8)
    p <- writeWKT(as(extent(b), 'SpatialPolygons'))
    q <- paste0("SELECT mukey, cokey, compname, compcpt_r FROM component
    WHERE mukey IN (SELECT DISTINCT mukey FROM

```

```

SDA_Get_Mukey_from_intersection_with_WktWgs84(' ', p,
      "') ORDER BY mukey, cokey, compct_r DESC")

x <- SDA_query(q)
str(x)
}
}

```

SDA\_spatialQuery

*SDA Spatial Query***Description**

Query SDA (SSURGO / STATSGO) records via spatial intersection with supplied geometries. Input can be SpatialPoints, SpatialLines, or SpatialPolygons objects with a valid CRS. Map unit keys, overlapping polygons, or the spatial intersection of geom + SSURGO / STATSGO polygons can be returned. See details.

**Usage**

```

SDA_spatialQuery(
  geom,
  what = "mukey",
  geomIntersection = FALSE,
  db = c("SSURGO", "STATSGO")
)

```

**Arguments**

geom	a Spatial* object, with valid CRS. May contain multiple features.
what	a character vector specifying what to return. 'mukey': data.frame with intersecting map unit keys and names, geom overlapping or intersecting map unit polygons
geomIntersection	logical; FALSE: overlapping map unit polygons returned, TRUE: intersection of geom + map unit polygons is returned.
db	a character vector identifying the Soil Geographic Databases ('SSURGO' or 'STATSGO') to query. Option <i>STATSGO</i> currently works only in combination with what = "geom".

**Details**

Queries for map unit keys are always more efficient vs. queries for overlapping or intersecting (i.e. least efficient) features. geom is converted to GCS / WGS84 as needed. Map unit keys are always returned when using what = "geom".

There is a 100,000 record limit and 32Mb JSON serializer limit, per query.

SSURGO (detailed soil survey, typically 1:24,000 scale) and STATSGO (generalized soil survey, 1:250,000 scale) data are stored together within SDA. This means that queries that don't specify an area symbol may result in a mixture of SSURGO and STATSGO records. See the examples below and the [SDA Tutorial](#) for details.

### Value

A `data.frame` if `what = 'mukey'`, otherwise `SpatialPolygonsDataFrame` object.

### Note

Row-order is not preserved across features in `geom` and returned object. Use `sp::over()` or similar functionality to extract from results. Polygon area in acres is computed server-side when `what = 'geom'` and `geomIntersection = TRUE`.

### Author(s)

D.E. Beaudette, A.G. Brown, D.R. Schlaepfer

### See Also

[SDA\\_query](#)

### Examples

```
if(requireNamespace("curl") &
    curl::has_internet() &
    requireNamespace("sp") &
    requireNamespace("raster")
) {

  library(aqp)
  library(sp)
  library(raster)

  ## query at a point

  # example point
  p <- SpatialPoints(cbind(x = -119.72330, y = 36.92204),
                     proj4string = CRS('+proj=longlat +datum=WGS84'))

  # query map unit records at this point
  res <- SDA_spatialQuery(p, what = 'mukey')

  # convert results into an SQL "IN" statement
  # useful when there are multiple intersecting records
  mu.is <- format_SQL_in_statement(res$mukey)

  # composite SQL WHERE clause
  sql <- sprintf("mukey IN %s", mu.is)
```



```

# inspect results
par(mar = c(0,0,3,1))
plot(ssurgo.geom, border = 'royalblue')
plot(statsgo.geom, lwd = 2, border = 'firebrick', add = TRUE)
plot(bbox.sp, lwd = 3, add = TRUE)
legend(
  x = 'topright',
  legend = c('BBOX', 'STATSGO', 'SSURGO'),
  lwd = c(3, 2, 1),
  col = c('black', 'firebrick', 'royalblue'),
)

# quick reminder that STATSGO map units often contain many components
# format an SQL IN statement using the first STATSGO mukey
mu.is <- format_SQL_in_statement(statsgo.geom$mukey[1])

# composite SQL WHERE clause
sql <- sprintf("mukey IN %s", mu.is)

# get commonly used map unit / component / chorizon records
# as a SoilProfileCollection object
x <- fetchSDA(sql)

# tighter figure margins
par(mar = c(0,0,3,1))

# organize component sketches by national map unit symbol
# color horizons via awc
# adjust legend title
# add alternate label (vertical text) containing component percent
# move horizon names into the profile sketches
# make profiles wider
groupedProfilePlot(
  x,
  groups = 'nationalmusym',
  label = 'compname',
  color = 'awc_r',
  col.label = 'Available Water Holding Capacity (cm / cm)',
  alt.label = 'compct_r',
  name.style = 'center-center',
  width = 0.3
)

mtext(
  'STATSGO (1:250,000) map units contain a lot of components!',
  side = 1,
  adj = 0,
  line = -1.5,
  at = 0.25,
  font = 4
)

```

```
)
}
```

---

seriesExtent

*Retrieve Soil Series Extent Maps from SoilWeb*


---

### Description

This function downloads a generalized representations of a soil series extent from SoilWeb, derived from the current SSURGO snapshot. Data can be returned as vector outlines (`SpatialPolygonsDataFrame` object) or gridded representation of area proportion falling within 800m cells (raster object). Gridded series extent data are only available in CONUS. Vector representations are returned with a GCS/WGS84 coordinate reference system and raster representations are returned with an Albers Equal Area / NAD83 coordinate reference system (EPSG 6350).

### Usage

```
seriesExtent(s, type = c("vector", "raster"), timeout = 60)
```

### Arguments

s	a soil series name, case-insensitive
type	series extent representation, vector results in a <code>SpatialPolygonsDataFrame</code> object and raster results in a raster object
timeout	time that we are willing to wait for a response, in seconds

### Note

This function requires the `rgdal` package.

### Author(s)

D.E. Beaudette

### References

<https://casoilresource.lawr.ucdavis.edu/see/>



**Examples**

```

if(requireNamespace("curl") &
  curl::has_internet()) {

  # required packages
  library(sp)
  library(raster)
  library(rgdal)

  # specify a soil series name
  s <- 'magnor'

  # return as SpatialPolygonsDataFrame
  x <- seriesExtent(s, type = 'vector')
  # return as raster
  y <- seriesExtent(s, type = 'raster')

  # note that CRS are different
  proj4string(x)
  projection(y)

  # transform vector representation to CRS of raster
  x <- spTransform(x, CRS(projection(y)))

  # graphical comparison
  par(mar = c(1, 1, 1, 3))
  plot(y, axes = FALSE)
  plot(x, add = TRUE)

}

```

---

siblings

*Lookup siblings and cousins for a given soil series.*


---

**Description**

Lookup siblings and cousins for a given soil series, from the current fiscal year SSURGO snapshot via SoilWeb.

**Usage**

```
siblings(s, only.major = FALSE, component.data = FALSE, cousins = FALSE)
```

**Arguments**

<code>s</code>	character vector, the name of a single soil series, case-insensitive.
<code>only.major</code>	logical, should only return siblings that are major components
<code>component.data</code>	logical, should component data for siblings (and optionally cousins) be returned?
<code>cousins</code>	logical, should siblings-of-siblings (cousins) be returned?

**Details**

The siblings of any given soil series are defined as those soil series (major and minor component) that share a parent map unit with the named series (as a major component). Cousins are siblings of siblings. Data are sourced from SoilWeb which maintains a copy of the current SSURGO snapshot.

**Value**

**sib** data.frame containing siblings, major component flag, and number of co-occurrences  
**sib.data** data.frame containing sibling component data  
**cousins** data.frame containing cousins, major component flag, and number of co-occurrences  
**cousin.data** data.frame containing cousin component data

**Author(s)**

D.E. Beaudette

**References**

- [Soil Series Query Functions](#)
- [Soil "Siblings" Tutorial](#)

**See Also**

[OSDquery](#), [siblings](#), [fetchOSD](#)

**Examples**

```
if(requireNamespace("curl") &
  curl::has_internet()) {

  # basic usage
  x <- siblings('zook')
  x$sib

  # restrict to siblings that are major components
  # e.g. the most likely siblings
  x <- siblings('zook', only.major = TRUE)
  x$sib
}
```

---

simplifyColorData      *Simplify Color Data by ID*

---

### Description

Simplify multiple Munsell color observations associated with each horizon.

### Usage

```
simplifyColorData(d, id.var = "phiid", wt = "colorpct", bt = FALSE)
```

### Arguments

d	a data.frame object, typically returned from NASIS, see details
id.var	character vector with the name of the column containing an ID that is unique among all horizons in d
wt	a character vector with the name of the column containing color weights for mixing
bt	logical, should the mixed sRGB representation of soil color be transformed to closest Munsell chips? This is performed by <code>aqp::rgb2Munsell</code> <code>aqp::rgb2Munsell</code>

### Details

This function is mainly intended for the processing of NASIS pedon/horizon data which may or may not contain multiple colors per horizon/moisture status combination. `simplifyColorData` will "mix" multiple colors associated with horizons in `d`, according to IDs specified by `id.var`, using "weights" (area percentages) specified by the `wt` argument to `mix_and_clean_colors`.

Note that this function doesn't actually simulate the mixture of pigments on a surface, rather, "mixing" is approximated via weighted average in the CIELAB colorspace.

The `simplifyColorData` function can be applied to data sources other than NASIS by careful use of the `id.var` and `wt` arguments. However, `d` must contain Munsell colors split into columns named "colorhue", "colorvalue", and "colorchroma". In addition, the moisture state ("Dry" or "Moist") must be specified in a column named "colormoistst".

The `mix_and_clean_colors` function can be applied to arbitrary data sources as long as `x` contains sRGB coordinates in columns named "r", "g", and "b". This function should be applied to chunks of rows within which color mixtures make sense.

Examples:

- [KSSL data](#)
- [soil color mixing tutorial](#)

### Author(s)

D.E. Beaudette

---

simplifyFragmentData *Simplify Coarse Fraction Data*

---

### Description

Simplify multiple coarse fraction (>2mm) records by horizon.

### Usage

```
simplifyFragmentData(rf, id.var, nullFragmentsAreZero = TRUE)
```

### Arguments

`rf` a data.frame object, typically returned from NASIS, see details

`id.var` character vector with the name of the column containing an ID that is unique among all horizons in `rf`

`nullFragmentsAreZero` should fragment volumes of NULL be interpreted as 0? (default: TRUE), see details

### Details

This function is mainly intended for the processing of NASIS pedon/horizon data which contains multiple coarse fragment descriptions per horizon. `simplifyFragmentData` will "sieve out" coarse fragments into the USDA classes, split into hard and para- fragments.

The `simplifyFragmentData` function can be applied to data sources other than NASIS by careful use of the `id.var` argument. However, `rf` must contain coarse fragment volumes in the column "fragvol", fragment size (mm) in columns "fragsize\_l", "fragsize\_r", "fragsize\_h", and fragment cementation class in "fraghard".

Examples:

- [KSSL data](#)

### Author(s)

D.E. Beaudette

---

*SoilWeb\_spatial\_query Get SSURGO Data via Spatial Query*

---

**Description**

Get SSURGO Data via Spatial Query to SoilWeb

**Usage**

```
SoilWeb_spatial_query(  
  bbox = NULL,  
  coords = NULL,  
  what = "mapunit",  
  source = "soilweb"  
)
```

**Arguments**

bbox	a bounding box in WGS84 geographic coordinates, see examples
coords	a coordinate pair in WGS84 geographic coordinates, see examples
what	data to query, currently ignored
source	the data source, currently ignored

**Details**

Data are currently available from SoilWeb. These data are a snapshot of the "official" data. The snapshot date is encoded in the "soilweb\_last\_update" column in the function return value. Planned updates to this function will include a switch to determine the data source: "official" data via USDA-NRCS servers, or a "snapshot" via SoilWeb.

**Value**

The data returned from this function will depend on the query style. See examples below.

**Note**

This function should be considered experimental; arguments, results, and side-effects could change at any time. SDA now supports spatial queries, consider using [SDA\\_query\\_features](#) instead.

**Author(s)**

D.E. Beaudette

**Examples**

```

if(requireNamespace("curl") &
  curl::has_internet()) {

  # query by bbox
  SoilWeb_spatial_query(bbox=c(-122.05, 37, -122, 37.05))

  # query by coordinate pair
  SoilWeb_spatial_query(coords=c(-121, 38))
}

```

---

STRplot

*Graphical Description of US Soil Taxonomy Soil Temperature Regimes*


---

**Description**

Graphical Description of US Soil Taxonomy Soil Temperature Regimes

**Usage**

```
STRplot(mast, msst, mwst, permafrost = FALSE, pt.cex = 2.75, leg.cex = 0.85)
```

**Arguments**

mast	single value or vector of mean annual soil temperature (deg C)
msst	single value or vector of mean summer soil temperature (deg C)
mwst	single value of mean winter soil temperature (deg C)
permafrost	logical: permafrost presence / absence
pt.cex	symbol size
leg.cex	legend size

**Details**

[Soil Temperature Regime Evaluation Tutorial](#)

**Author(s)**

D.E. Beaudette

**References**

Soil Survey Staff. 2015. Illustrated guide to soil taxonomy. U.S. Department of Agriculture, Natural Resources Conservation Service, National Soil Survey Center, Lincoln, Nebraska.

**See Also**[estimateSTR](#)**Examples**

```
par(mar=c(4,1,0,1))
STRplot(mast = 0:25, msst = 10, mwst = 1)
```

---

taxaExtent

*Retrieve Soil Taxonomy Membership Grids*


---

**Description**

This function downloads a generalized representation of the geographic extent of any single taxon from the top 4 tiers of Soil Taxonomy, or those taxa matching a given formative element used in greatgroup or subgroup taxa. Data are provided by SoilWeb, ultimately sourced from the current SSURGO snapshot. Data are returned as raster objects representing area proportion falling within 800m cells. Currently area proportions are based on major components only. Data are only available in CONUS and returned using an Albers Equal Area / NAD83 coordinate reference system (EPSG 6350).

**Usage**

```
taxaExtent(
  x,
  level = c("order", "suborder", "greatgroup", "subgroup"),
  formativeElement = FALSE,
  timeout = 60
)
```

**Arguments**

x	single taxon label (e.g. haploxera1fs) or formative element (e.g. pale), case-insensitive
level	the taxonomic level within the top 4 tiers of Soil Taxonomy, one of c('order', 'suborder', 'greatgroup', 'subgroup')
formativeElement	logical, search using formative elements instead of taxon label
timeout	time that we are willing to wait for a response, in seconds

## Details

### Taxon Queries:

Taxon labels can be conveniently extracted from the "ST\_unique\_list" sample data, provided by the [SoilTaxonomy package](#).

### Formative Element Queries:

*Greatgroup::*

The following labels are used to access taxa containing the following formative elements (in parenthesis).

- acr: (acro/acr) extreme weathering
- alb: (alb) presence of an albic horizon
- anhy: (anhy) very dry
- anthra: (anthra) presence of an anthropic epipedon
- aqu: (aqui/aqu) wetness
- argi: (argi) presence of an argillic horizon
- calci: (calci) presence of a calcic horizon
- cry: (cryo/cry) cryic STR
- dur: (duri/dur) presence of a duripan
- dystr: (dystro/dystr) low base saturation
- endo: (endo) ground water table
- epi: (epi) perched water table
- eutr: (eutro/eutr) high base saturation
- ferr: (ferr) presence of Fe
- fibr: (fibr) least decomposed stage
- fluv: (fluv) flood plain
- fol: (fol) mass of leaves
- fragi: (fragi) presence of a fragipan
- fragloss: (fragloss) presence of a fragipan and glossic horizon
- frasi: (frasi) not salty
- fulv: (fulvi/fulv) dark brown with organic carbon
- glac: (glac) presence of ice lenses
- gloss: (glosso/gloss) presence of a glossic horizon
- gypsi: (gypsi) presence of a gypsic horizon
- hal: (hal) salty
- hemi: (hemi) intermediate decomposition
- hist: (histo/hist) organic soil material
- hum: (humi/hum) presence of organic carbon
- hydr: (hydro/hydr) presence of water
- kandi: (kandi) presence of a kandic horizon
- kanhap: (kanhaplo/kanhap) thin kandic horizon
- luvi: (luvi) illuvial organic material
- melan: (melano/melan) presence of a melanic epipedon



- moll: (molli/moll) presence of a mollic epipedon
- natr: (natri/natr) presence of a natric horizon
- pale: (pale) excessive development
- petr: (petro/petr) petrocalcic horizon
- plac: (plac) presence of a thin pan
- plagg: (plagg) presence of a plaggen epipedon
- plinth: (plinth) presence of plinthite
- psamm: (psammo/psamm) sandy texture
- quartz: (quartz) high quartz content
- rhod: (rhodo/rhod) dark red colors
- sal: (sali/sal) presence of a salic horizon
- sapr: (sapr) most decomposed stage
- sombri: (sombri) presence of a sombric horizon
- sphagno: (sphagno) presence of sphagnum moss
- sulf: (sulfo/sulfi/sulf) presence of sulfides or their oxidation products
- torri: (torri) torric/aridic SMR
- ud: (udi/ud) udic SMR
- umbr: (umbri/umbr) presence of an umbric epipedon
- ust: (usti/ust) ustic SMR
- verm: (verm) wormy, or mixed by animals
- vitr: (vitri/vitr) presence of glass
- xer: (xero/xer) xeric SMR

*Subgroup:*

The following labels are used to access taxa containing the following formative elements (in parenthesis).

- abruptic: (abruptic) abrupt textural change
- acric: (acric) low apparent CEC
- aeric: (aeric) more aeration than typic subgroup
- albaquic: (albaquic) presence of albic minerals, wetter than typic subgroup
- albic: (albic) presence of albic minerals
- alfic: (alfic) presence of an argillic or kandic horizon
- alic: (alic) high extractable Al content
- anionic: (anionic) low CEC or positively charged
- anthraquic: (anthraquic) human controlled flooding as in paddy rice culture
- anthropic: (anthropic) an anthropic epipedon
- aquic: (aquic) wetter than typic subgroup
- arenic: (arenic) 50-100cm sandy textured surface
- argic: (argic) argillic horizon
- aridic: (aridic) more aridic than typic subgroup
- calcic: (calcic) presence of a calcic horizon
- chromic: (chromic) high chroma colors
- cumulic: (cumulic) thickened epipedon

- duric: (duric) presence of a duripan
- durinodic: (durinodic) presence of durinodes
- dystric: (dystric) lower base saturation percentage
- entic: (entic) minimal surface/subsurface development
- eutric: (eutric) higher base saturation percentage
- fibric: (fibric) >25cm of fibric material
- fluvaquentic: (fluvaquentic) wetter than typical subgroup, evidence of stratification
- fragiaquic: (fragiaquic) presence of fragic properties, wetter than typical subgroup
- fragic: (fragic) presence of fragic properties
- glacic: (glacic) presence of ice lenses or wedges
- glossaquic: (glossaquic) interfingering horizon boundaries, wetter than typical subgroup
- glossic: (glossic) interfingering horizon boundaries
- grossarenic: (grossarenic) >100cm sandy textured surface
- gypsic: (gypsic) presence of gypsic horizon
- halic: (halic) salty
- haplic: (haplic) central theme of subgroup concept
- hemic: (hemic) >25cm of hemic organic material
- humic: (humic) higher organic matter content
- hydric: (hydric) presence of water
- kandic: (kandic) low activity clay present
- lamellic: (lamellic) presence of lamellae
- leptic: (leptic) thinner than typical subgroup
- limnic: (limnic) presence of a limnic layer
- lithic: (lithic) shallow lithic contact present
- natric: (natric) presence of sodium
- nitric: (nitric) presence of nitrate salts
- ombroaquic: (ombroaquic) surface wetness
- oxyaquic: (oxyaquic) water saturated but not reduced
- pachic: (pachic) epipedon thicker than typical subgroup
- petrocalcic: (petrocalcic) presence of a petrocalcic horizon
- petroferric: (petroferric) presence of petroferric contact
- petrogypsic: (petrogypsic) presence of a petrogypsic horizon
- petronodic: (petronodic) presence of concretions and/or nodules
- placic: (placic) presence of a placic horizon
- plinthic: (plinthic) presence of plinthite
- rhodic: (rhodic) darker red colors than typical subgroup
- ruptic: (ruptic) intermittent horizon
- salic: (salic) presence of a salic horizon
- sapric: (sapric) >25cm of sapric organic material
- sodic: (sodic) high exchangeable Na content
- sombric: (sombic) presence of a sombric horizon
- sphagnic: (sphagnic) sphagnum organic material

- sulfic: (sulfic) presence of sulfides
- terric: (terric) mineral substratum within 1 meter
- thapto: (thaptic/thapto) presence of a buried soil horizon
- turbic: (turbic) evidence of cryoturbation
- udic: (udic) more humid than typic subgroup
- umbric: (umbric) presence of an umbric epipedon
- ustic: (ustic) more ustic than typic subgroup
- vermic: (vermic) animal mixed material
- vitric: (vitric) presence of glassy material
- xanthic: (xanthic) more yellow than typic subgroup
- xeric: (xeric) more xeric than typic subgroup

### Value

a raster object

### Author(s)

D.E. Beaudette and Andrew Brown

### Examples

```
if(requireNamespace("curl") &
  curl::has_internet()) {

  library(raster)

  # try a couple of different examples

  # soil order
  taxa <- 'vertisols'
  x <- taxaExtent(taxa, level = 'order')
  a <- raster::aggregate(x, fact = 5)

  # suborder
  taxa <- 'ustalfs'
  x <- taxaExtent(taxa, level = 'suborder')
  a <- raster::aggregate(x, fact = 5)

  # greatgroup
  taxa <- 'haplohumults'
  x <- taxaExtent(taxa, level = 'greatgroup')
  a <- raster::aggregate(x, fact = 5)

  # subgroup
  taxa <- 'Typic Haploxerepts'
  x <- taxaExtent(taxa, level = 'subgroup')
  a <- raster::aggregate(x, fact = 5)
```

```

# greatgroup formative element
taxa <- 'psamm'
x <- taxaExtent(taxa, level = 'greatgroup', formativeElement = TRUE)
a <- raster::aggregate(x, fact = 5)

# subgroup formative element
taxa <- 'abruptic'
x <- taxaExtent(taxa, level = 'subgroup', formativeElement = TRUE)
a <- raster::aggregate(x, fact = 5)

# quick evaluation of the result
if(requireNamespace("rasterVis") & requireNamespace('viridisLite')) {
  rasterVis::levelplot(a,
    margin = FALSE, scales = list(draw = FALSE),
    col.regions = viridisLite::viridis,
    main = names(a)
  )
}

# slippy map
if(requireNamespace("mapview")) {
  mapview::mapview(a, col.regions = viridisLite::viridis, na.color = NA, use.layer.names = TRUE)
}

}

```

---

 unicode

---

*Convert coded values returned from NASIS and SDA queries to factors*


---

### Description

These functions convert the coded values returned from NASIS or SDA to factors (e.g. 1 = Alfisols) using the metadata tables from NASIS. For SDA the metadata is pulled from a static snapshot in the soilDB package (/data/metadata.rda).

### Usage

```

unicode(
  df,
  invert = FALSE,
  db = "NASIS",
  droplevels = FALSE,
  stringsAsFactors = default.stringsAsFactors(),

```

```

    dsn = NULL
  )

```

### Arguments

<code>df</code>	data.frame
<code>invert</code>	converts the code labels back to their coded values (FALSE)
<code>db</code>	label specifying the soil database the data is coming from, which indicates whether or not to query metadata from local NASIS database ("NASIS") or use soilDB-local snapshot ("LIMS" or "SDA")
<code>droplevels</code>	logical: indicating whether to drop unused levels in classifying factors. This is useful when a class has large number of unused classes, which can waste space in tables and figures.
<code>stringsAsFactors</code>	logical: should character vectors be converted to factors?
<code>dsn</code>	Optional: path to local SQLite database containing NASIS table structure; default: NULL

### Details

These functions convert the coded values returned from NASIS into their plain text representation. It duplicates the functionality of the CODELABEL function found in NASIS. This function is primarily intended to be used internally by other soilDB R functions, in order to minimize the need to manually convert values.

The function works by iterating through the column names in a data frame and looking up whether they match any of the ColumnPhysicalNames found in the metadata domain tables. If matches are found then the columns coded values are converted to their corresponding factor levels. Therefore it is not advisable to reuse column names from NASIS unless the contents match the range of values and format found in NASIS. Otherwise `uncode()` will convert their values to NA.

When data is being imported from NASIS, the metadata tables are sourced directly from NASIS. When data is being imported from SDA or the NASIS Web Reports, the metadata is pulled from a static snapshot in the soilDB package.

Beware the default is to return the values as factors rather than strings. While strings are generally preferable, factors make plotting more convenient. Generally the factor level ordering returned by `uncode()` follows the naturally ordering of categories that would be expected (e.g. sand, silt, clay).

### Value

A data frame with the results.

### Author(s)

Stephen Roecker

## Examples

```
if(requireNamespace("curl") &
  curl::has_internet() &
  require(aqp)) {
  # query component by nationalmusym
  comp <- fetchSDA(WHERE = "nationalmusym = '2vzcp'")
  s <- site(comp)

  # use SDA uncoding domain via db argument
  s <- uncode(s, db="SDA")
  levels(s$taxorder)
}
```

---

us\_ss\_timeline

*Timeline of US Published Soil Surveys*

---

## Description

This dataset contains the years of each US Soil Survey was published.

## Format

A data.frame with 5209 observations on the following 5 variables.

- "ssa": Soil Survey name, a character vector
- "year": Year of publication, a numeric vector
- "pdf": Does a manuscript PDF document exist? a logical vector
- "state": State abbreviation, a character vector

## Details

This data was web scraped from the NRCS Soils Website. The scraping procedure and a example plot are included in the examples section below.

## Source

<https://www.nrcs.usda.gov/wps/portal/nrcs/soilsurvey/soils/survey/state/>

---

waterDayYear	<i>Compute Water Day and Year</i>
--------------	-----------------------------------

---

### Description

Compute "water" day and year, based on the end of the typical or legal dry season. This is September 30 in California.

### Usage

```
waterDayYear(d, end = "09-30")
```

### Arguments

d	anything that can be safely converted to POSIXlt
end	"MM-DD" notation for end of water year

### Details

This function doesn't know about leap-years. Probably worth checking.

### Value

A data.frame object with the following

wy	the "water year"
wd	the "water day"

### Author(s)

D.E. Beaudette

### References

Ideas borrowed from: <https://github.com/USGS-R/dataRetrieval/issues/246> and <https://stackoverflow.com/questions/48123049/create-day-index-based-on-water-year>

### Examples

```
# try it
waterDayYear('2019-01-01')
```

---

`WCS_details`*Web Coverage Services Details*

---

**Description**

List variables or databases provided by soilDB web coverage service (WCS) abstraction. These lists will be expanded in future versions.

**Usage**

```
WCS_details(wcs = c("mukey", "ISSR800"))
```

**Arguments**

`wcs` a WCS label ('mukey' or 'ISSR800')

**Value**

a `data.frame`

**Examples**

```
WCS_details(wcs = 'ISSR800')
```



# Index

- \* **IO**
  - parseWebReport, [93](#)
- \* **datasets**
  - loafercreek, [83](#)
  - NASIS\_table\_column\_keys, [91](#)
  - SCAN\_SNOTEL\_metadata, [97](#)
  - us\_ss\_timeline, [118](#)
- \* **hplot**
  - STRplot, [110](#)
- \* **manip**
  - estimateSTR, [8](#)
  - fetchGDB, [10](#)
  - fetchHenry, [11](#)
  - fetchNASISLabData, [17](#)
  - fetchNASISWebReport, [18](#)
  - fetchOSD, [21](#)
  - fetchPedonPC, [24](#)
  - fetchSCAN, [26](#)
  - fetchSDA, [28](#)
  - get\_colors\_from\_NASIS\_db, [38](#)
  - get\_colors\_from\_pedon\_db, [38](#)
  - get\_comonth\_from\_NASIS\_db, [39](#)
  - get\_component\_data\_from\_NASIS\_db, [40](#)
  - get\_cosoilmoist\_from\_NASIS, [41](#)
  - get\_extended\_data\_from\_NASIS\_db, [44](#)
  - get\_extended\_data\_from\_pedon\_db, [45](#)
  - get\_hz\_data\_from\_NASIS\_db, [46](#)
  - get\_hz\_data\_from\_pedon\_db, [47](#)
  - get\_lablayer\_data\_from\_NASIS\_db, [48](#)
  - get\_labpedon\_data\_from\_NASIS\_db, [49](#)
  - get\_site\_data\_from\_NASIS\_db, [73](#)
  - get\_site\_data\_from\_pedon\_db, [74](#)
  - get\_soilseries\_from\_NASIS, [75](#)
  - get\_text\_notes\_from\_NASIS\_db, [75](#)
  - get\_veg\_data\_from\_NASIS\_db, [76](#)
  - get\_veg\_from\_AK\_Site, [77](#)
  - get\_veg\_from\_MT\_veg\_db, [78](#)
  - get\_veg\_from\_NPS\_PLOTS\_db, [79](#)
  - get\_veg\_other\_from\_MT\_veg\_db, [79](#)
  - get\_veg\_species\_from\_MT\_veg\_db, [80](#)
  - OSDquery, [91](#)
  - SDA\_query, [98](#)
  - SDA\_spatialQuery, [100](#)
  - siblings, [105](#)
  - simplifyColorData, [107](#)
  - simplifyFragmentData, [108](#)
  - SoilWeb\_spatial\_query, [109](#)
  - unicode, [116](#)
  - waterDayYear, [119](#)
- \* **package**
  - soilDB-package, [3](#)
- \* **utilities**
  - fetchKSSL, [13](#)
  - fetchRaCA, [25](#)
  - .dump\_NASIS\_table, [4](#)
  - .get\_NASIS\_fkey\_by\_name
    - (.get\_NASIS\_table\_key\_by\_name), [4](#)
  - .get\_NASIS\_pkey\_by\_name
    - (.get\_NASIS\_table\_key\_by\_name), [4](#)
  - .get\_NASIS\_pkeyref\_by\_name
    - (.get\_NASIS\_table\_key\_by\_name), [4](#)
  - .get\_NASIS\_table\_key\_by\_name, [4](#)
  - .get\_NASIS\_table\_name\_by\_purpose, [5](#)
- code (unicode), [116](#)
- createStaticNASIS, [6](#)
- dbConnectNASIS, [7](#)
- dbQueryNASIS, [7](#)
- estimateColorMixture, [8](#)

- estimateSTR, 8, *111*
- fetchGDB, 10
- fetchHenry, 11
- fetchKSSL, 13
- fetchNASIS, 4, 15, 40–42
- fetchNASISLabData, 17
- fetchNASISWebReport, 18
- fetchOSD, 14, 21, 26, 92, 106
- fetchPedinPC, 4, 24
- fetchRaCA, 25
- fetchSCAN, 13, 26
- fetchSDA, 28
- fetchSDA\_spatial, 31
- fetchSoilGrids, 33
- fetchVegdata, 34
- filter\_geochem, 35
- format\_SQL\_in\_statement, 36
- get\_chorizon\_from\_NASISWebReport (fetchNASISWebReport), 18
- get\_chorizon\_from\_SDA (fetchSDA), 28
- get\_cointerp\_from\_SDA (fetchSDA), 28
- get\_colors\_from\_NASIS\_db, 38
- get\_colors\_from\_pedin\_db, 38, 47
- get\_comonth\_from\_NASIS\_db, 39
- get\_component\_cogeomorph\_data\_from\_NASIS\_db (fetchNASIS), 15
- get\_component\_copm\_data\_from\_NASIS\_db (fetchNASIS), 15
- get\_component\_correlation\_data\_from\_NASIS\_db (fetchNASIS), 15
- get\_component\_data\_from\_NASIS\_db, 40
- get\_component\_diaghz\_from\_NASIS\_db (fetchNASIS), 15
- get\_component\_esd\_data\_from\_NASIS\_db (fetchNASIS), 15
- get\_component\_from\_GDB (fetchGDB), 10
- get\_component\_from\_NASISWebReport (fetchNASISWebReport), 18
- get\_component\_from\_SDA (fetchSDA), 28
- get\_component\_horizon\_data\_from\_NASIS\_db (fetchNASIS), 15
- get\_component\_otherveg\_data\_from\_NASIS\_db (fetchNASIS), 15
- get\_component\_restrictions\_from\_NASIS\_db (get\_component\_data\_from\_NASIS\_db), 40
- get\_concentrations\_from\_NASIS\_db (fetchNASIS), 15
- get\_copedon\_from\_NASIS\_db (fetchNASIS), 15
- get\_cosoilmoist\_from\_NASIS, 41
- get\_cosoilmoist\_from\_NASISWebReport, 42
- get\_cosoilmoist\_from\_NASISWebReport (fetchNASISWebReport), 18
- get\_cosoilmoist\_from\_SDA, 42
- get\_cosoilmoist\_from\_SDA (fetchSDA), 28
- get\_cotext\_from\_NASIS\_db (fetchNASIS), 15
- get\_EDIT\_ecoclass\_by\_geoUnit, 43
- get\_extended\_data\_from\_NASIS\_db, 44
- get\_extended\_data\_from\_pedin\_db, 45
- get\_hz\_data\_from\_NASIS\_db, 38, 44, 46, 46, 74
- get\_hz\_data\_from\_pedin\_db, 25, 39, 45, 47, 74, 76, 78
- get\_lablayer\_data\_from\_NASIS\_db, 48, 49
- get\_labpedon\_data\_from\_NASIS\_db, 18, 48, 49
- get\_legend\_from\_GDB (fetchGDB), 10
- get\_legend\_from\_NASIS (fetchVegdata), 34
- get\_legend\_from\_NASISget\_lmuaoverlap\_from\_NASIS (fetchNASIS), 15
- get\_legend\_from\_NASISWebReport (fetchNASISWebReport), 18
- get\_legend\_from\_SDA (fetchSDA), 28
- get\_lmuaoverlap\_from\_NASIS (fetchVegdata), 34
- get\_lmuaoverlap\_from\_NASISWebReport (fetchNASISWebReport), 18
- get\_lmuaoverlap\_from\_SDA (fetchSDA), 28
- get\_mapunit\_from\_GDB (fetchGDB), 10
- get\_mapunit\_from\_NASIS (fetchNASIS), 15
- get\_mapunit\_from\_NASISWebReport (fetchNASISWebReport), 18
- get\_mapunit\_from\_SDA (fetchSDA), 28
- get\_mutext\_from\_NASIS\_db (fetchNASIS), 15
- get\_NOAA\_GHCND, 50
- get\_NOAA\_stations\_nearXY, 51
- get\_OSD\_JSON, 52
- get\_phfmp\_from\_NASIS\_db (fetchNASIS), 15
- get\_phorizon\_from\_NASIS\_db (fetchNASIS), 15

- get\_progress\_from\_NASISWebReport  
(fetchNASISWebReport), 18
- get\_project\_correlation\_from\_NASISWebReport  
(fetchNASISWebReport), 18
- get\_project\_from\_NASISWebReport  
(fetchNASISWebReport), 18
- get\_projectmapunit2\_from\_NASISWebReport  
(fetchNASISWebReport), 18
- get\_projectmapunit\_from\_NASIS  
(fetchNASIS), 15
- get\_projectmapunit\_from\_NASISWebReport  
(fetchNASISWebReport), 18
- get\_RMF\_from\_NASIS\_db (fetchNASIS), 15
- get\_SDA\_hydric, 53
- get\_SDA\_interpretation, 53
- get\_SDA\_muaggatt, 69
- get\_SDA\_pmgrouppname, 70
- get\_SDA\_property, 70
- get\_site\_data\_from\_NASIS\_db, 38, 44, 46,  
73
- get\_site\_data\_from\_pedon\_db, 39, 45, 47,  
74, 76, 78
- get\_sitesoilmoist\_from\_NASISWebReport  
(fetchNASISWebReport), 18
- get\_soilseries\_from\_NASIS, 75
- get\_soilseries\_from\_NASISWebReport  
(get\_soilseries\_from\_NASIS), 75
- get\_text\_notes\_from\_NASIS\_db, 75
- get\_veg\_data\_from\_NASIS\_db, 76
- get\_veg\_from\_AK\_Site, 74, 77
- get\_veg\_from\_MT\_veg\_db, 78, 80
- get\_veg\_from\_NPS\_PLOTS\_db, 79
- get\_veg\_other\_from\_MT\_veg\_db, 78, 79, 80
- get\_veg\_species\_from\_MT\_veg\_db, 78, 80,  
80
- get\_vegplot\_from\_NASIS\_db  
(fetchVegdata), 34
- get\_vegplot\_location\_from\_NASIS\_db  
(fetchVegdata), 34
- get\_vegplot\_species\_from\_NASIS\_db  
(fetchVegdata), 34
- get\_vegplot\_textnote\_from\_NASIS\_db  
(fetchVegdata), 34
- get\_vegplot\_transect\_from\_NASIS\_db  
(fetchVegdata), 34
- get\_vegplot\_transpecies\_from\_NASIS\_db  
(fetchVegdata), 34
- get\_vegplot\_tree\_si\_details\_from\_NASIS\_db  
(fetchVegdata), 34
- get\_vegplot\_tree\_si\_summary\_from\_NASIS\_db  
(fetchVegdata), 34
- get\_vegplot\_trhi\_from\_NASIS\_db  
(fetchVegdata), 34
- getHzErrorsNASIS, 37
- getHzErrorsPедonPC, 24
- getHzErrorsPедonPC (fetchPедonPC), 24
- gopheridge (loafercreek), 83
- ISSR800.wcs, 81
- KSSL\_VG\_model, 82
- loafercreek, 4, 83
- local\_NASIS\_defined, 84
- make\_EDIT\_service\_URL, 86
- makeChunks, 85
- mapunit\_geom\_by\_ll\_bbox, 88, 98
- metadata (unicode), 116
- mineralKing (loafercreek), 83
- mix\_and\_clean\_colors, 89
- mixMunsell, 8
- month2season (fetchHenry), 11
- mukey.wcs, 90
- NASIS (dbConnectNASIS), 7
- NASIS\_table\_column\_keys, 91
- OSDquery, 23, 91, 106
- parseWebReport, 93
- processSDA\_WKT, 94
- ROSETTA, 95
- SCAN\_sensor\_metadata (fetchSCAN), 26
- SCAN\_site\_metadata (fetchSCAN), 26
- SCAN\_SNOTEL\_metadata, 97
- SDA\_make\_spatial\_query  
(SDA\_spatialQuery), 100
- SDA\_query, 4, 29, 98, 101
- SDA\_query\_features, 109
- SDA\_query\_features (SDA\_spatialQuery),  
100
- SDA\_spatialQuery, 100
- seriesExtent, 104
- siblings, 23, 92, 105, 106
- simplifyFragmentData  
(simplifyFragmentData), 108

simplifyArtifactData  
    (simplifyFragmentData), [108](#)  
simplifyColorData, [38](#), [107](#)  
simplifyFragmentData, [108](#)  
soilDB (soilDB-package), [3](#)  
soilDB-package, [3](#)  
soilDB.env (soilDB-package), [3](#)  
SoilWeb\_spatial\_query, [109](#)  
state\_FIPS\_codes  
    (SCAN\_SNOTEL\_metadata), [97](#)  
STRplot, [9](#), [110](#)  
summarizeSoilTemperature (fetchHenry),  
    [11](#)  
  
taxaExtent, [111](#)  
  
uncode, [116](#)  
us\_ss\_timeline, [118](#)  
  
waterDayYear, [119](#)  
WCS\_details, [120](#)