Package 'soilDB'

July 23, 2025

Type Package

Title Soil Database Interface

Version 2.8.11

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

Description A collection of functions for reading soil data from U.S. Department of Agriculture Natural Resources Conservation Service (USDA-NRCS) and National Cooperative Soil Survey (NCSS) databases.

License GPL (>= 3)

LazyLoad yes

Depends R (>= 3.5.0)

Imports grDevices, graphics, stats, utils, methods, aqp (>= 2.1.0), data.table, DBI, curl

Suggests jsonlite, xml2, httr, rvest, odbc, RSQLite, sf, wk, terra, raster, knitr, rmarkdown, testthat

Repository CRAN

URL https://github.com/ncss-tech/soilDB/, https://ncss-tech.github.io/soilDB/, https://ncss-tech.github.io/AQP/

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

RoxygenNote 7.3.2

Encoding UTF-8

Language en-US

LazyData false

VignetteBuilder knitr

NeedsCompilation no

Author Dylan Beaudette [aut], Jay Skovlin [aut], Stephen Roecker [aut], Andrew Brown [aut, cre] (ORCID: <https://orcid.org/0000-0002-4565-533X>)

Date/Publication 2025-07-11 03:40:02 UTC

Contents

soilDB-package . . . 4
createSSURGO . . . 5
createStaticNASIS . . . 7
dbConnectNASIS . . . 8
dbQueryNASIS . . . 8
downloadSSURGO . . . 9
estimateColorMixture . . . 10
estimateSTR . . . 11
fetchHWSD . . . 12
fetchKSSL . . . 14
fetchLDM . . . 16
fetchNASIS . . . 19
fetchNASISLabData . . . 21
fetchNASISWebReport . . . 22
fetchOSD . . . 24
fetchPedonPC . . . 26
fetchRaCA . . . 27
fetchSCAN . . . 29
fetchSDA_spatial . . . 32
fetchSoilGrids . . . 34
fetchSOLUS . . . 37
fetchSRI . . . 40
fetchVegdata . . . 41
filter_geochem . . . 44
format_SQL_in_statement . . . 45
getHzErrorsNASIS . . . 45
get_colors_from_NASIS_db . . . 46
get_colors_from_pedon_db . . . 47
get_comonth_from_NASIS_db . . . 48
get_component_data_from_NASIS_db . . . 49
get_component_from_GDB . . . 51
get_component_from_SDA . . . 53
get_cosoilmoist_from_NASIS . . . 55
get_ecosite_history_from_NASIS_db . . . 57
get_EDIT_ecoclass_by_geoUnit . . . 57
get_extended_data_from_NASIS_db . . . 58
get_extended_data_from_pedon_db . . . 59
get_hz_data_from_NASIS_db . . . 60
get_hz_data_from_pedon_db . . . 61
get_lablayer_data_from_NASIS_db . . . 61
get_labpedon_data_from_NASIS_db . . . 62
get_mapunit_from_NASIS . . . 63
get_NASIS_metadata . . . 64
get_NASIS_table_key_by_name . . . 65
get_NASIS_table_metadata . . . 66
get_NASIS_table_name_by_purpose . . . 67

get_NOAA_GHCND . . . 68
get_NOAA_stations_nearXY . . . 69
get_OSD . . . 70
get_phroots_from_NASIS_db . . . 71
get_projectmilestone_from_NASIS . . . 71
get_RMF_from_NASIS_db . . . 72
get_SDA_coecoclass . . . 73
get_SDA_cosurfmorph . . . 74
get_SDA_hydric . . . 76
get_SDA_interpretation . . . 78
get_SDA_metrics . . . 95
get_SDA_muaggatt . . . 96
get_SDA_pmgroupname . . . 97
get_SDA_property . . . 98
get_SDV_legend_elements . . . 101
get_site_data_from_NASIS_db . . . 102
get_site_data_from_pedon_db . . . 104
get_soilseries_from_NASIS . . . 104
get_SRI . . . 106
get_SRI_layers . . . 108
get_text_notes_from_NASIS_db . . . 109
get_veg_data_from_NASIS_db . . . 110
get_veg_from_AK_Site . . . 111
get_veg_from_MT_veg_db . . . 111
get_veg_from_NPS_PLOTS_db . . . 112
get_veg_other_from_MT_veg_db . . . 113
get_veg_species_from_MT_veg_db . . . 113
ISSR800.wcs . . . 114
KSSL_VG_model . . . 115
loafercreek . . . 117
local_NASIS_defined . . . 118
makeChunks . . . 119
make_EDIT_service_URL . . . 120
metadata . . . 122
mukey.wcs . . . 122
NASISChoiceList . . . 124
NASISDomainsAsFactor . . . 125
NASISLocalDatabase . . . 126
NASIS_table_column_keys . . . 127
OSDquery . . . 127
parseWebReport . . . 129
processSDA_WKT . . . 130
ROSETTA . . . 131
SCAN_SNOTEL_metadata . . . 133
SDA_query . . . 133
SDA_spatialQuery . . . 135
seriesExtent . . . 139
siblings . . . 141

simplifyArtifactData . . . 143
simplifyColorData . . . 144
soilColor.wcs . . . 145
soilDB.env . . . 146
soilDB_user_dir . . . 147
SoilWeb_spatial_query . . . 148
STRplot . . . 149
summarizeSoilTemperature . . . 150
taxaExtent . . . 152
uncode . . . 157
us_ss_timeline . . . 158
waterDayYear . . . 159
WCS_details . . . 160

Index **161**

`soilDB-package` *Soil Database Interface*

Description

A collection of functions for reading soil data from U.S. Department of Agriculture Natural Resources Conservation Service (USDA-NRCS) and National Cooperative Soil Survey (NCSS) databases

Details

This package provides methods for extracting soils information from local NASIS databases (MS SQL Server), local PedonPC and AKSite databases (MS Access format), Soil Data Access, and other soil-related web services.

Author(s)

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

See Also

`fetchNASIS`, `SDA_query`, `loafercreek`

createSSURGO *Create a database from SSURGO Exports*

Description

The following database types are tested and fully supported:

- SQLite or Geopackage
- DuckDB
- Postgres or PostGIS

Usage

```
createSSURGO(
  filename = NULL,
  exdir,
  conn = NULL,
  pattern = NULL,
  include_spatial = TRUE,
  include_tabular = TRUE,
  dissolve_field = NULL,
  maxruledepth = 0,
  overwrite = FALSE,
  header = FALSE,
  quiet = TRUE,
  ...
)
```

Arguments

filename	*character*. Output file name (e.g. 'db.sqlite' or 'db.gpkg'). Only used when con is not specified by the user.
exdir	*character*. Path containing containing input SSURGO spatial (.shp) and tabular (.txt) files, downloaded and extracted by downloadSSURGO() or similar.
conn	A *DBIConnection* object. Default is a SQLiteConnection used for writing .sqlite or .gpkg files. Alternate options are any DBI connection types. When include_spatial=TRUE, the sf package is used to write spatial data to the database.
pattern	*character*. Optional regular expression to use to filter subdirectories of exdir. Default: NULL will search all subdirectories for SSURGO export files.
include_spatial	*logical* or *character*. Include spatial data layers in database? Default: TRUE inserts all spatial tables. If include_spatial is a *character* vector containing table names, only that set are written to file. e.g. include_spatial=c("mupolygon", "featpoint") writes only the mapunit polygons and special feature points.

include_tabular
: *logical* or *character*. Include tabular data layers in database? Default: TRUE inserts all tabular tables. If include_tabular is a *character* vector containing table names, only that set are written to file. e.g. include_tabular=c("mapunit", "muaggatt") writes only the mapunit and muaggatt tables. Note that special feature descriptions are stored in table "featdesc" and metadata for each soil survey area are stored in "soil_metadata" tables.

dissolve_field
: *character*. Dissolve geometries to create MULTIPOLYGON features? Column name specified is the grouping variable. Default: NULL does no aggregation, giving 1 POLYGON feature per delineation. "mukey" aggregates all related delineations within a soil survey area.

maxruledepth
: *integer*. Maximum rule depth for "cointerp" table. Default 0 includes only shallowest ratings for smaller database size.

overwrite
: *logical*. Overwrite existing layers? Default FALSE will append to existing tables/layers.

header
: *logical*. Passed to read.delim() for reading pipe-delimited (|) text files containing tabular data.

quiet
: *logical*. Suppress messages and other output from database read/write operations?

...
: Additional arguments passed to write_sf() for writing spatial layers.

Details

In theory any other DBI-compatible data source can be used for output. See conn argument. If you encounter issues using specific DBI connection types, please report in the soilDB issue tracker.

Value

Character. Vector of layer/table names in filename.

See Also

downloadSSURGO()

Examples

```
## Not run:
 downloadSSURGO("areasymbol IN ('CA067', 'CA077', 'CA632')", destdir = "SSURGO_test")
 createSSURGO("test.gpkg", "SSURGO_test")

## 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 DBI::dbListTables 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 "_View_1"). Default: TRUE
dsn	Optional: path to SQLite database containing NASIS table structure; or a DBIConnection. 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 *Create local NASIS database connection*

Description

Create a connection to a local NASIS database with DBI

Usage

```
dbConnectNASIS(dsn = NULL)

NASIS(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 *Query a NASIS DBIConnection*

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

downloadSSURGO *Get SSURGO ZIP files from Web Soil Survey 'Download Soils Data'*

Description

Download ZIP files containing spatial (ESRI shapefile) and tabular (TXT) files with standard SSURGO format; optionally including the corresponding SSURGO Template Database with `include_template=TRUE`.

Usage

```
downloadSSURGO(
  WHERE = NULL,
  areasymbols = NULL,
  destdir = tempdir(),
  exdir = destdir,
  include_template = FALSE,
  db = c("SSURGO", "STATSGO"),
  extract = TRUE,
  remove_zip = FALSE,
  overwrite = FALSE,
  quiet = FALSE
)
```

Arguments

`WHERE`	*character*. A SQL `WHERE` clause expression used to filter records in `sacatalog` table. Alternately `WHERE` can be any spatial object supported by `SDA_spatialQuery()` for defining the target extent.
`areasymbols`	*character*. Character vector of soil survey area symbols e.g. `c("CA067", "CA077")`. Used in lieu of `WHERE` argument.
`destdir`	*character*. Directory to download ZIP files into. Default `tempdir()`.
`exdir`	*character*. Directory to extract ZIP archives into. May be a directory that does not yet exist. Each ZIP file will extract to a folder labeled with `areasymbol` in this directory. Default: `destdir`
`include_template`	*logical*. Include the (possibly state-specific) MS Access template database? Default: `FALSE`
`db`	*character*. Either `"SSURGO"` (default; detailed soil map) or `"STATSGO"` (general soil map).
`extract`	*logical*. Extract ZIP files to `exdir`? Default: `TRUE`
`remove_zip`	*logical*. Remove ZIP files after extracting? Default: `FALSE`
`overwrite`	*logical*. Overwrite by re-extracting if directory already exists? Default: `FALSE`
`quiet`	*logical*. Passed to `curl::curl_download()`.

Details

To specify the Soil Survey Areas you would like to obtain data you use a `WHERE` clause for query of `sacatalog` table such as `areasymbol = 'CA067'`, `"areasymbol IN ('CA628', 'CA067')"` or `areasymbol LIKE 'CT%'`.

When `db="STATSGO"` the `WHERE` argument is not supported. Allowed `areasymbols` include `"US"` and two-letter state codes e.g. `"WY"` for the Wyoming general soils map.

Pipe-delimited TXT files are found in */tabular/* folder extracted from a SSURGO ZIP. The files are named for tables in the SSURGO schema. There is no header / the files do not have column names. See the *Soil Data Access Tables and Columns Report*: https://sdmdataaccess.nrcs.usda.gov/documents/TablesAndColumnsReport.pdf for details on tables, column names and metadata including the default sequence of columns used in TXT files. The function returns a `try-error` if the `WHERE`/`areasymbols` arguments result in

Several ESRI shapefiles are found in the */spatial/* folder extracted from a SSURGO ZIP. These have prefix `soilmu_` (mapunit), `soilsa_` (survey area), `soilsf_` (special features). There will also be a TXT file with prefix `soilsf_` describing any special features. Shapefile names then have an `a_` (polygon), `l_` (line), `p_` (point) followed by the soil survey area symbol.

Value

character. Paths to downloaded ZIP files (invisibly). May not exist if `remove_zip = TRUE`.

See Also

`createSSURGO()`

`estimateColorMixture` *Estimate color mixtures using weighted average of CIELAB color coordinates*

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`	numeric. fractional weights, usually area of horizon face
`backTransform`	logical, should the mixed sRGB representation of soil color be transformed to closest Munsell chips? This is performed by `aqp::col2Munsell()` default: `FALSE`

Value

A data.frame containing estimated color mixture

Note

See `aqp::mixMunsell()` for a more realistic (but slower) simulation of subtractive mixing of pigments. An efficient replacement for this function (wt. mean in CIELAB coordinates) is implemented in `aqp::mixMunsell(..., mixingMethod = 'estimate')`.

Author(s)

D.E. Beaudette

`estimateSTR` *Estimate Soil Temperature Regime*

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)
```

---

fetchHWSD *Fetch Harmonized World Soil Database Data*

Description

Creates a local cache of FAO Harmonized World Soil Database (HWSD) information. Source raster map in ESRI Grid format (.bil) is converted to GeoTIFF. The source tabular database in Microsoft Access (.mdb) format is converted to SQLite.

Usage

```
fetchHWSD(
  x = NULL,
 hwsd_url = "https://s3.eu-west-1.amazonaws.com/data.gaezdev.aws.fao.org/HWSD/",
  hwsd_version = 2L,
  force = FALSE
)

get_HWSD_path(what = c("sqlite", "mdb", "raster", "path"), hwsd_version = 2L)
```

Arguments

x	A *SpatRaster*, *SpatVector*, *SpatExtent*, or any other object that has a *SpatExtent*. Default NULL returns the full dataset. Extent of interest that is passed to terra::crop() y argument.
hwsd_url	*character*. URL for downloading HWSD dataset. Default "https://s3.eu-west-1.amazonaws.com/da
hwsd_version	*integer*. Default 2L.
force	Force rebuilding of cache. Default: FALSE
what	*character*. One of "sqlite", "raster", or "path"

Value

A *SpatRaster* object with mapunit-level aggregate information stored as categories.

Source

Food and Agriculture Organization of the United Nations (FAO), Soils Portal, Harmonized World Soil Database (HWSD) v2.0 https://www.fao.org/soils-portal/data-hub/soil-maps-and-databases/harmonized-world-soil-database-v20/en/

Examples

```
## Not run:
  x <- terra::vect(system.file("ex", "lux.shp", package = "terra"))
  res <- fetchHWSD(x)

  # categorical data (WRB class)
  terra::activeCat(res) <- "WRB4"

  # view WRB4 map
  terra::plot(res)
  terra::lines(x, col = "white")

  # convert categories containing numeric data to numeric values
  res2 <- terra::catalyze(res)

  # view AWC map
  terra::plot(res2$AWC, main = "Available Water Capacity, mm")
  terra::lines(x, col = "white")

  # access tabular data from cached SQLite database
  SDA_query("SELECT * FROM HWSD2_SMU LIMIT 1", dsn = get_HWSD_path())

## End(Not run)
```

fetchKSSL *Get Kellogg Soil Survey Laboratory Data from SoilWeb snapshot*

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 interface has largely been superseded by the Soil Data Access snapshot of the Laboratory Data Mart, available via `fetchLDM()`.

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://ncsslabdatamart.sc.egov.usda.gov/

See Also

`fetchOSD`

Examples

```
library(aqp)

# 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
aqp::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
```

`fetchLDM` *Query data from Kellogg Soil Survey Laboratory Data Mart via Soil Data Access or local SQLite snapshot*

Description

This function provides access to the Kellogg Soil Survey Laboratory Data Mart via Soil Data Access or a local SQLite snapshot. See details and examples for additional usage instructions.

Usage

```
fetchLDM(
  x = NULL,
  what = "pedlabsampnum",
  bycol = "pedon_key",
  tables = c("lab_physical_properties", "lab_chemical_properties",
   "lab_calculations_including_estimates_and_default_values", "lab_rosetta_Key"),
  WHERE = NULL,
  chunk.size = 1000,
  ntries = 3,
  layer_type = c("horizon", "layer", "reporting layer"),
```

```
  area_type = c("ssa", "country", "state", "county", "mlra", "nforest", "npark"),
  prep_code = c("S", ""),
  analyzed_size_frac = c("<2 mm", ""),
  dsn = NULL
)
```

Arguments

x	A vector of values to find in column specified by what, default NULL uses no constraints on what
what	A single column name from tables: lab_combine_nasis_ncss, lab_webmap, lab_site, lab_pedon or lab_area. Common choices include pedlabsampnum (Laboratory Pedon ID), upedonid (User Pedon ID), corr_name ('Correlated' Taxon Name), samp_name ('Sampled As' Taxon Name), or area_code (area symbol for specified lab_area records, see area_type).
bycol	A single column name from lab_layer used for processing chunks; default: "pedon_key"
tables	A vector of table names; Default is "lab_physical_properties", "lab_chemical_properties", "lab_calculations_including_estimates_and_default_values", and "lab_rosetta_Key". May also include one or more of: "lab_mir", "lab_mineralogy_glass_count", "lab_major_and_trace_elements_and_oxides", "lab_xray_and_thermal" but it will be necessary to select appropriate prep_code and analyzed_size_frac for your analysis (see *Details*).
WHERE	character. A custom SQL WHERE clause, which overrides x, what, and bycol, such as CASE WHEN corr_name IS NOT NULL THEN LOWER(corr_name) ELSE LOWER(samp_name) END =
chunk.size	Number of pedons per chunk (for queries that may exceed maxJsonLength)
ntries	Number of tries (times to halve chunk.size) before returning NULL; default 3
layer_type	Default: "horizon", "layer", and "reporting layer"
area_type	Default: "ssa" (Soil Survey Area). Other options include (choose one): "country", "state", "county", "mlra" (Major Land Resource Area), "nforest" (National Forest), "npark" (National Park)
prep_code	Default: "S" and "". May also include one or more of: "F", "HM", "HM_SK" "GP", "M", "N", or "S"
analyzed_size_frac	Default: "<2 mm" and "". May also include one or more of: "<0.002 mm", "0.02-0.05 mm", "0.05-0.1 mm", "0.1-0.25 mm", "0.25-0.5 mm", "0.5-1 mm", "1-2 mm", "0.02-2 mm", "0.05-2 mm"
dsn	Data source name; either a path to a SQLite database, an open DBIConnection or (default) NULL (to use soilDB::SDA_query)

Details

You can download SQLite or GeoPackage snapshots here: https://ncsslabdatamart.sc.egov.usda.gov/database_download.aspx. Specify the dsn argument to use a local copy of the lab data rather than Soil Data Access web service.

Lab Data Mart model diagram: https://jneme910.github.io/Lab_Data_Mart_Documentation/Documents/SDA_KSSL_Data_model.html If the chunk.size parameter is set too large and the Soil Data Access request fails, the algorithm will re-try the query with a smaller (halved) chunk.size argument. This will be attempted up to 3 times before returning NULL

The default behavior joins the lab_area tables only for the "Soil Survey Area" related records. You can specify alternative area records for use in x, what or WHERE arguments by setting area_type to a different value.

When requesting data from "lab_major_and_trace_elements_and_oxides", "lab_mineralogy_glass_count", or "lab_xray_and_thermal" multiple preparation codes (prep_code) or size fractions (analyzed_size_frac) are possible. The default behavior of fetchLDM() is to attempt to return a topologically valid (minimal overlaps) *SoilProfileCollection*. This is achieved by setting prep_code="S" ("sieved") and analyzed_size_frac="<2 mm". You may specify alternate or additional preparation codes or fractions as needed, but note that this may cause "duplication" of some layers where measurements were made with different preparation or on fractionated samples

Value

a SoilProfileCollection for a successful query, a try-error if no site/pedon locations can be found or NULL for an empty lab_layer (within sites/pedons) result

Examples

```
## Not run:

  # fetch by Soil Survey Area area symbol (area_code using default "ssa" area_type)
  res <- fetchLDM("CA630", what = "area_code")

  # fetch by Major Land Resource area symbol (area_code using "mlra" area_type)
  res <- fetchLDM("22A", what = "area_code", area_type = "mlra")

  # fetch by multiple case-insensitive taxon name
  # (correlated or sampled as Musick or Holland series)
  res <- fetchLDM(WHERE = "(CASE WHEN corr_name IS NOT NULL
                                THEN LOWER(corr_name)
                                ELSE LOWER(samp_name)
                           END) IN ('musick', 'holland')")

  # physical properties of soils correlated as taxonomic subgroup "Typic Argialbolls"
  res <- fetchLDM(x = "Typic Argialbolls",
                  what = "corr_taxsubgrp",
                  tables = "lab_physical_properties")


## End(Not run)
```

`fetchNASIS` *Get a pedon or component data* `SoilProfileCollection` *from NASIS*

Description

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

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 manageable data frames. Primarily these functions access the local NASIS database using an ODBC connection. The `dsn` argument allows you to specify a path or `DBIConnection` to an SQLite database. The argument `from = "pedon_report"`, data can be read from the NASIS Report 'fetchNASIS', from either text file or URL (specified as `url`). The primary purpose of `fetchNASIS(from = "pedon_report")` is importing datasets larger than 8000+ pedons/components.

Tutorials:

- fetchNASIS Columns
- fetchNASIS Pedons Tutorial
- fetchNASIS Components Tutorial

Usage

```
fetchNASIS(
  from = "pedons",
  url = NULL,
  SS = TRUE,
  rmHzErrors = FALSE,
  nullFragsAreZero = TRUE,
  soilColorState = "moist",
  mixColors = FALSE,
  lab = FALSE,
  fill = FALSE,
  dropAdditional = TRUE,
  dropNonRepresentative = TRUE,
  duplicates = FALSE,
  stringsAsFactors = NULL,
  dsn = NULL
)

get_concentrations_from_NASIS_db(
  SS = TRUE,
  stringsAsFactors = NULL,
  dsn = NULL
)

get_phfmp_from_NASIS_db(SS = TRUE, stringsAsFactors = NULL, dsn = NULL)
```

Arguments

from	Determines what objects should fetched? Default: 'pedons'. Alternately, 'components', or '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: FALSE)
nullFragsAreZero	Should fragment volumes of NULL be interpreted as 0? (default: TRUE), see details
soilColorState	Used only for from = 'pedons'; which colors should be used to generate the convenience field soil_color? ('moist' or 'dry')
mixColors	Should mixed colors be calculated where multiple colors are populated for the same moisture state in a horizon? Default FALSE takes the dominant color for each horizon moist/dry state.
lab	Should the phlabresults child table be fetched with site/pedon/horizon data (default: FALSE)
fill	Include pedon or component records without horizon data in result? (default: FALSE)
dropAdditional	Used only for from='components' with duplicates = TRUE. Prevent "duplication" of mustatus == "additional" mapunits? Default: TRUE
dropNonRepresentative	Used only for from='components' with duplicates = TRUE. Prevent "duplication" of non-representative data mapunits? Default: TRUE
duplicates	Used only for from='components'. Duplicate components for all instances of use (i.e. one for each legend data mapunit is used on; optionally for additional mapunits, and/or non-representative data mapunits?). This will include columns from get_component_correlation_data_from_NASIS_db() that identify which legend(s) a component is used on.
stringsAsFactors	deprecated
dsn	Optional: path or *DBIConnection* to local database containing NASIS table structure; default: NULL

Details

The value of nullFragsAreZero will have a significant impact on the rock fragment fractions returned by fetchNASIS. Set nullFragsAreZero = FALSE in those cases where there are many data-gaps and NULL rock fragment values should be interpreted as NULL. Set nullFragsAreZero = 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. Pedon IDs that are missing horizon data, or have errors in their horizonation 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.

Value

A SoilProfileCollection object

Author(s)

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

See Also

get_component_data_from_NASIS()

---

fetchNASISLabData *Get NCSS Pedon laboratory data from NASIS*

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

Value

a SoilProfileCollection object

Author(s)

J.M. Skovlin and D.E. Beaudette

See Also

get_labpedon_data_from_NASIS_db

`fetchNASISWebReport` *Get component tables from NASIS Web Reports*

Description

Get component tables from NASIS Web Reports

Usage

```
fetchNASISWebReport(
  projectname,
  rmHzErrors = FALSE,
  fill = FALSE,
  stringsAsFactors = NULL
)

get_component_from_NASISWebReport(projectname, stringsAsFactors = NULL)

get_chorizon_from_NASISWebReport(
  projectname,
  fill = FALSE,
  stringsAsFactors = NULL
)

get_legend_from_NASISWebReport(
  mlraoffice,
  areasymbol,
  droplevels = TRUE,
  stringsAsFactors = NULL
)

get_lmuaoverlap_from_NASISWebReport(
  areasymbol,
  droplevels = TRUE,
  stringsAsFactors = NULL
)

get_mapunit_from_NASISWebReport(
  areasymbol,
  droplevels = TRUE,
  stringsAsFactors = NULL
)

get_projectmapunit_from_NASISWebReport(projectname, stringsAsFactors = NULL)

get_projectmapunit2_from_NASISWebReport(
  mlrassoarea,
```

```
  fiscalyear,
  projectname,
  stringsAsFactors = NULL
)

get_project_from_NASISWebReport(mlrassoarea, fiscalyear)

get_progress_from_NASISWebReport(mlrassoarea, fiscalyear, projecttypename)

get_project_correlation_from_NASISWebReport(
  mlrassoarea,
  fiscalyear,
  projectname
)

get_cosoilmoist_from_NASISWebReport(
  projectname,
  impute = TRUE,
  stringsAsFactors = NULL
)

get_sitesoilmoist_from_NASISWebReport(usiteid)
```

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	deprecated
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)

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)
usiteid	character: User Site IDs

Value

A data.frame or list with the results.

Author(s)

Stephen Roecker

---

fetchOSD *Get Official Series Descriptions and summaries from SoilWeb API*

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 updated quarterly and made available via SoilWeb. Set extended = TRUE and see the soilweb.metadata list element for information on when the source data were last updated.

Usage

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

Arguments

soils	a character vector of named soil series; case-insensitive
colorState	color state for horizon soil color visualization: "moist" or "dry"
extended	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

mtnpos 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

shape_across empirical probabilities for surface shape (across-slope) from the current SSURGO snapshot

shape_down empirical probabilities for surface shape (down-slope) 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

ecoclassid area cross-tabulation of ecoclassid by soil series name, derived from the current SSURGO snapshot, major components only

climate climate summaries from PRISM stack (CONUS only)

NCCPI select quantiles of NCCPI and Irrigated NCCPI, derived from the current SSURGO snapshot

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 summary tables (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 that uses DAYMET.
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.

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.

Note

Requests to the SoilWeb API are split into batches of 100 series names from `soils` via `makeChunks()`.

Author(s)

D.E. Beaudette, A.G. Brown

References

USDA-NRCS OSD search tools: https://soilseries.sc.egov.usda.gov/

See Also

OSDquery(), siblings()

Examples

```
library(aqp)
# 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
par(mar=c(0,0,0,0), mfrow=c(2,1))
aqp::plotSPC(
  s.moist,
  name = 'hzname',
  cex.names = 0.85,
  depth.axis = list(line = -4)
)
aqp::plotSPC(
  s.dry,
  name = 'hzname',
  cex.names = 0.85,
  depth.axis = list(line = -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))
aqp::plotSPC(x$SPC)
str(x, 1)
```

---

fetchPedonPC *Get a SoilProfileCollection 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)

getHzErrorsPedonPC(dsn, strict = TRUE)
```

Arguments

dsn	The path to a PedonPC version 6.x database
strict	Use "strict" horizon error checking? Default: TRUE

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 horizonation 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 horizonation. 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

NOTICE: The SoilWeb snapshot of the RaCA data has been deprecated. The latest version of the data, including values measured by the Kellogg Soil Survey Laboratory, and supporting documentation, are available here: https://www.nrcs.usda.gov/resources/data-and-reports/rapid-carbon-assessment-raca. Download link on National Agricultural Library Ag Data Commons: https://data.nal.usda.gov/dataset/rapid-carbon-assessment-raca

Get Rapid Carbon Assessment (RaCA) data by state, geographic bounding-box, RaCA site ID, or soil series query from the SoilWeb API. This interface to the data was an experimental delivery service that does not include the latest soil organic carbon (SOC) measurements.

Please use current RaCA distribution if you need lab *measured* SOC rather than SOC estimated by VNIR.

This interface will be updated sometime calendar year 2022 to include the latest soil morphology, taxonomic classification, and measured SOC values. More detailed coordinates for sample sites should also be available.

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://data.nal.usda.gov/dataset/rapid-carbon-assessment-raca

See Also

fetchOSD

`fetchSCAN` *Get Daily Climate Data from USDA-NRCS SCAN (Soil Climate Analysis Network) Stations*

Description

Query soil/climate data from USDA-NRCS SCAN Stations.

Usage

```
fetchSCAN(
  site.code = NULL,
  year = NULL,
  report = "SCAN",
  timeseries = c("Daily", "Hourly"),
  tz = "US/Central",
  ...
)

SCAN_sensor_metadata(site.code)

SCAN_site_metadata(site.code = NULL)
```

Arguments

`site.code`	a vector of site codes. If `NULL` `SCAN_site_metadata()` returns metadata for all SCAN sites and no sensor data.
`year`	a vector of years
`report`	report name, single value only; default `'SCAN'`, other example options include individual sensor codes, e.g. `'SMS'` for Soil Moisture Storage, `'TEMP'` for temperature
`timeseries`	either `'Daily'` or `'Hourly'`
`tz`	Target timezone to convert `datetime` columns of results. Default: `"US/Central"`.
`...`	additional arguments. May include `intervalType`, `format`, `sitenum`, `interval`, `year`, `month`. Presence of additional arguments bypasses default batching functionality provided in the function and submits a 'raw' request to the API form.

Details

Possible above and below ground sensor types include: 'SMS' (soil moisture), 'STO' (soil temperature), 'SAL' (salinity), 'TAVG' (daily average air temperature), 'TMIN' (daily minimum air temperature), 'TMAX' (daily maximum air temperature), 'PRCP' (daily precipitation), 'PREC' (daily precipitation), 'SNWD' (snow depth), 'WTEQ' (snow water equivalent),'WDIRV' (wind direction), 'WSPDV' (wind speed), 'LRADT' (solar radiation/langley total).

This function converts below-ground sensor depth from inches to cm. All temperature values are reported as degrees C. Precipitation, snow depth, and snow water content are reported as *inches*.

The `datetime` column in sensor data results is converted to the target time zone specified in `tz` argument, the default is `"US/Central"`. Use `tz = "UTC"` (or other `OlsonNames()` that do not use daylight savings, e.g. `"US/Arizona"`) to avoid having a mix of time offsets due to daylight savings time.

SCAN Sensors:

All Soil Climate Analysis Network (SCAN) sensor measurements are reported hourly.

Element Measured	Sensor Type
Air Temperature	Shielded thermistor
Barometric Pressure	Silicon capacitive pressure sensor
Precipitation	Storage-type gage or tipping bucket
Relative Humidity	Thin film capacitance-type sensor
Snow Depth	Sonic sensor (not on all stations)
Snow Water Content	Snow pillow device and a pressure transducer (not on all stations)
Soil Moisture	Dielectric constant measuring device. Typical measurements are at 2", 4", 8", 20", and 40" where
Soil Temperature	Encapsulated thermistor. Typical measurements are at 2", 4", 8", 20", and 40" where possible.
Solar Radiation	Pyranometer
Wind Speed and Direction	Propellor-type anemometer

SNOTEL Sensors:

All Snow Telemetry (SNOTEL) sensor measurements are reported daily.

Element Measured	Sensor Type
Air Temperature	Shielded thermistor
Barometric Pressure	Silicon capacitive pressure sensor
Precipitation	Storage-type gage or tipping bucket
Relative Humidity	Thin film capacitance-type sensor
Snow Depth	Sonic sensor
Snow Water Content	Snow pillow device and a pressure transducer
Soil Moisture	Dielectric constant measuring device. Typical measurements are at 2", 4", 8", 20", and 40" where
Soil Temperature	Encapsulated thermistor. Typical measurements are at 2", 4", 8", 20", and 40" where possible.
Solar Radiation	Pyranometer
Wind Speed and Direction	Propellor-type anemometer

See the fetchSCAN tutorial for additional usage and visualization examples.

Value

a `list` of `data.frame` objects, where each element name is a sensor type, plus a `metadata` table; different `report` types change the types of sensor data returned. `SCAN_sensor_metadata()` and `SCAN_site_metadata()` return a `data.frame`. `NULL` on bad request.

Author(s)

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

References

See the Soil Climate Analysis Network home page for more information on the SCAN program, and links to other associated programs such as SNOTEL, at the National Weather and Climate Center. You can get information on available web services, as well as interactive maps of snow water equivalent, precipitation and streamflow.

Examples

```
## Not run:
    # get data
    x <- try(fetchSCAN(site.code = c(356, 2072), year = c(2015, 2016)))
    str(x, 1)

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

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

    # # get hourly data (warning, result is large ~11MB)
    # x <- try(fetchSCAN(site.code = c(356, 2072),
    #                    year = 2015,
    #                    timeseries = "Hourly"))
    #
   # # data are in US/Central time, standard or daylight savings time based on day of year
    # unique(format(x$SMS$datetime, '%Z'))
    #
    # # the site metadata indicate timeseries data time zone (dataTimeZone)
    # # for site 356 the timezone is offset of 8 hours behind UTC
    #
    # # to obtain all datetime data with a consistent offset use ETC GMT offset
    # # e.g. "Etc/GMT+8". note the sign is inverted ("GMT+8" vs. `dataTimeZone=-8`)
    # x <- try(fetchSCAN(site.code = c(356, 2072),
    #          year = 2015,
    #          timeseries = "Hourly",
    #          tz = "Etc/GMT+8"))


## End(Not run)
```

fetchSDA_spatial *Get Spatial Data from Soil Data Access by* mukey*,* nationalmusym *or* areasymbol

Description

This method facilitates queries to Soil Data Access (SDA) mapunit and survey area geometry. Queries are generated based on map unit key (mukey) and national map unit symbol (nationalmusym) for mupolygon (SSURGO) or gsmmupolygon (STATSGO) geometry OR legend key (lkey) and area symbols (areasymbol) for sapolygon (Soil Survey Area; SSA) geometry).

A Soil Data Access query returns geometry and key identifying information about the map unit or area of interest. Additional columns from the map unit or legend table can be included; see add.fields argument.

Usage

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

Arguments

x	A vector of map unit keys (mukey) or national map unit symbols (nationalmusym) for mupolygon, muline or mupoint; feature keys (featkey) for featpoint and featline; legend keys (lkey) or soil survey area symbols (areasymbol) for sapolygon geometry. If geom.src="mlrapolygon" then x refers to MLRARSYM (major land resource area symbols).
by.col	Column name containing map unit identifier "mukey", "nationalmusym", or "ecoclassid" for geom.src mupolygon OR "areasymbol", "areaname", "mlraoffice", "mouagncyresp" for geom.src sapolygon; default is determined by isTRUE(is.numeric(x)) for mukey, featkey or lkey, using nationalmusym or areasymbol otherwise.
method	geometry result type: "feature" returns polygons, "bbox" returns the bounding box of each polygon (via STEnvelope()), "point" returns a single point (via STPointOnSurface()) within each polygon, "extent" returns an aggregate bounding box (the extent of all polygons, geometry::EnvelopeAggregate()), "convexhull" (geometry::ConvexHullAggregate()) returns the aggregate convex hull around all polygons, "union" (geometry::UnionAggregate()) and "collection" (geometry::CollectionAggregate()) return a MULTIPOLYGON

	or a GEOMETRYCOLLECTION, respectively, for each mukey, nationalmusym, or areasymbol . In the case of the latter four aggregation methods, the groups for aggregation depend on by.col (default by "mukey").
geom.src	Either mupolygon (map unit polygons), muline (map unit lines), mupoint (map unit points), featpoint (feature points), featline (feature lines), sapolygon (soil survey area boundary polygons), or mlrapolygon (major land resource area boundary polygons)
db	Default: "SSURGO". When geom.src is mupolygon, use STATSGO polygon geometry instead of SSURGO by setting db = "STATSGO"
add.fields	Column names from mapunit or legend table to add to result. Must specify parent table name as the prefix before column name e.g. mapunit.muname.
chunk.size	Number of values of x to process per query. Necessary for large results. Default: 10
verbose	Print messages?
as_Spatial	Return sp classes? e.g. Spatial*DataFrame. Default: FALSE.

Details

This function automatically "chunks" the input vector (using makeChunks()) of map unit 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 map units 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.

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

A prototype interface, geom.src="mlrapolygon", is provided for obtaining Major Land Resource Area (MLRA) polygon boundaries. When using this geometry source x is a vector of MLRARSYM (MLRA Symbols). The geometry source is the MLRA Geographic Database v5.2 (2022) which is not (yet) part of Soil Data Access. Instead of SDA, GDAL utilities are used to read a zipped ESRI Shapefile from a remote URL: https://www.nrcs.usda.gov/sites/default/files/2022-10/MLRA_52_2022.zip. Therefore, most additional fetchSDA_spatial() arguments are *not* currently supported for the MLRA geometry source. In the future a mlrapolygon table may be added to SDA (analogous to mupolygon and sapolygon), and the function will be updated accordingly at that time.

Value

an sf data.frame corresponding to SDA spatial data for all symbols requested. If as_Spatial=TRUE returns a Spatial*DataFrame from the sp package via sf::as_Spatial() for backward compatibility. 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, Dylan E. Beaudette

Examples

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

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

# compare extent of nmusym to single mukey within it
if (!inherits(single.mukey, 'try-error') &&
    !inherits(full.extent.nmusym, 'try-error')) {

    if (requireNamespace("sf")) {

     plot(sf::st_geometry(full.extent.nmusym), col = "RED", border = 0)
     plot(sf::st_geometry(single.mukey), add = TRUE, col = "BLUE", border = 0)

    }

}

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

`fetchSoilGrids` *Get SoilGrids 2.0 Property Estimates for Points or Spatial Extent*

Description

This function obtains SoilGrids 2.0 properties information (250m raster resolution) given a `data.frame` containing site IDs, latitudes and longitudes, or a spatial extent (see `grid=TRUE` argument).

SoilGrids API and maps return values as whole (integer) numbers to minimize the storage space used. These values have conversion factors applied by `fetchSoilGrids()` to produce conventional units shown in the table below (see Details).

Usage

```
fetchSoilGrids(
  x,
  loc.names = c("id", "lat", "lon"),
  depth_intervals = c("0-5", "5-15", "15-30", "30-60", "60-100", "100-200"),
```

```
  variables = c("bdod", "cec", "cfvo", "clay", "nitrogen", "phh2o", "sand", "silt",
    "soc", "ocd", "wv0010", "wv0033", "wv1500"),
  grid = FALSE,
  filename = NULL,
  overwrite = TRUE,
  target_resolution = c(250, 250),
  summary_type = c("Q0.05", "Q0.5", "Q0.95", "mean"),
  endpoint = ifelse(!grid, "https://rest.isric.org/soilgrids/v2.0/properties/query",
    "https://files.isric.org/soilgrids/latest/data/"),
  ...,
  verbose = FALSE,
  progress = FALSE
)
```

Arguments

x	A `data.frame` containing 3 columns referring to site ID, latitude and longitude. Or a spatial (sf, terra) object for which a bounding box can be calculated when `grid=TRUE`.
loc.names	Optional: Column names referring to site ID, latitude and longitude. Default: `c("id", "lat", "lon")`
depth_intervals	Default: `"0-5"`, `"5-15"`, `"15-30"`, `"30-60"`, `"60-100"`, `"100-200"`
variables	Default: `"bdod"`, `"cec"`, `"cfvo"`, `"clay"`, `"nitrogen"`, `"phh2o"`, `"sand"`, `"silt"`, `"soc"`, `"ocd"`, `"wv0010"`, `"wv0033"`, `"wv1500"`. Optionally `"ocs"` (only for 0 to 30 cm interval).
grid	Download subset of SoilGrids Cloud Optimized GeoTIFF? Default: `FALSE`
filename	Only used when `grid=TRUE`. If `NULL` defaults to an in-memory raster, or temporary file if result does not fit in memory.
overwrite	Only used when `grid=TRUE`. Default: `FALSE`
target_resolution	Only used when `grid=TRUE`. Default: `c(250, 250)` (250m x 250m pixels)
summary_type	Only used when `grid=TRUE`. One or more of `"Q0.05"`, `"Q0.5"`, `"Q0.95"`, `"mean"`; these are summary statistics that correspond to 5th, 50th, 95th percentiles, and mean value for selected `variables`.
endpoint	Optional: custom API endpoint. Default: `"https://rest.isric.org/soilgrids/v2.0/properties/q` when `grid=FALSE`; `"https://files.isric.org/soilgrids/latest/data/"` when `grid=TRUE`.
...	Additional arguments passed to `terra::writeRaster()` when `grid=TRUE`.
verbose	Print messages? Default: `FALSE`
progress	logical, give progress when iterating over multiple requests; Default: `FALSE`

Details

Properties:

Name	Description	Mapped units
bdod	Bulk density of the fine earth fraction	cg/cm^3
cec	Cation Exchange Capacity of the soil	mmol(c)/kg
cfvo	Volumetric fraction of coarse fragments (> 2 mm)	cm^3/dm^3 (vol per mil)
clay	Proportion of clay particles (< 0.002 mm) in the fine earth fraction	g/kg
nitrogen	Total nitrogen (N)	cg/kg
phh2o	Soil pH	pH*10
sand	Proportion of sand particles (> 0.05 mm) in the fine earth fraction	g/kg
silt	Proportion of silt particles (>= 0.002 mm and <= 0.05 mm) in the fine earth fraction	g/kg
soc	Soil organic carbon content in the fine earth fraction	dg/kg
ocd	Organic carbon density	hg/m^3
ocs	Organic carbon stocks (0-30cm depth interval only)	t/ha
wv0010	Volumetric Water Content at 10kPa	0.1 v% or 1 mm/m
wv0033	Volumetric Water Content at 33kPa	0.1 v% or 1 mm/m
wv1500	Volumetric Water Content at 1500kPa	0.1 v% or 1 mm/m

SoilGrids predictions are made for the six standard depth intervals specified in the GlobalSoilMap IUSS working group and its specifications. The default depth intervals returned are (in centimeters): `"0-5"`, `"5-15"`, `"15-30"`, `"30-60"`, `"60-100"`, `"100-200"` for the properties `"bdod"`, `"cec"`, `"cfvo"`, `"clay"`, `"nitrogen"`, `"phh2o"`, `"sand"`, `"silt"`, `"soc"`, `"ocd"`, `"wv0010"`, `"wv0033"`, `"wv1500"`–each with percentiles (5th, 50th, 95th), mean and uncertainty values. The summary statistic name will be appended to the abbreviate variable name for each depth interval returned. Soil organic carbon stocks (0-30cm) (`variables="ocs"`) are returned only for `depth_intervals="0-30"`. The uncertainty values are the ratio between the inter-quantile range (90% prediction interval width) and the median : `(Q0.95-Q0.05)/Q0.50`. All values are converted from "mapped" to "conventional" based on above table conversion factors. Point data requests are made through `"properties/query"` endpoint of the SoilGrids v2.0 REST API. Please check ISRIC's data policy, disclaimer and citation: https://www.isric.org/about/data-policy.

Find out more information about the SoilGrids and GlobalSoilMap products here:

- https://www.isric.org/explore/soilgrids/faq-soilgrids
- https://www.isric.org/sites/default/files/GlobalSoilMap_specifications_december_2015_2.pdf

Value

A *SoilProfileCollection* (or *SpatRaster* when `grid=TRUE`). Returns `try-error` if all requests fail. Any error messages resulting from parsing will be echoed when `verbose=TRUE`.

Author(s)

Andrew G. Brown

References

- **Common soil chemical and physical properties:** Poggio, L., de Sousa, L. M., Batjes, N. H., Heuvelink, G. B. M., Kempen, B., Ribeiro, E., and Rossiter, D.: SoilGrids 2.0: producing soil

information for the globe with quantified spatial uncertainty, SOIL, 7, 217–240, 2021. DOI: doi:10.5194/soil72172021

- **Soil water content at different pressure heads:** Turek, M.E., Poggio, L., Batjes, N. H., Armindo, R. A., de Jong van Lier, Q., de Sousa, L.M., Heuvelink, G. B. M. : Global mapping of volumetric water retention at 100, 330 and 15000 cm suction using the WoSIS database, International Soil and Water Conservation Research, 11-2, 225-239, 2023. DOI: doi:10.1016/j.iswcr.2022.08.001

Examples

```
## Not run:
  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 <- try(fetchSoilGrids(your.points))

  if (!inherits(x, 'try-error'))
   aqp::plotSPC(x, name = NA, color = "socQ50")

  # organic carbon stocks use 0-30cm interval
  y <- try(fetchSoilGrids(your.points[1, ],
                          depth_interval = c("0-5", "0-30", "5-15", "15-30"),
                          variables = c("soc", "bdod", "ocd", "ocs")))

  # extract horizons from a SoilProfileCollection where horizon 2 overlaps 1, 3, and 4
  h <- aqp::horizons(y)

  # "ocs" (organic carbon stock 0-30cm interval)
  h[2, ]

  h$thickness_meters <- ((h$hzdepb - h$hzdept) / 100)

 # estimate "ocs" from modeled organic carbon and bulk density in 0-5, 5-15, 15-30 intervals
  #  (sum the product of soc, bdod, and thickness in meters)
  #  (1 gram per cubic decimeter = 1 kilogram per cubic meter)
  sum(h$socmean * h$bdodmean * h$thickness_meters, na.rm = TRUE)

  # estimate "ocs" from modeled organic carbon density in 0-5, 5-15, 15-30 intervals
  #  (sum the product of "ocd" and thickness in meters)
  sum(h$ocdmean * h$thickness_meters, na.rm = TRUE)


## End(Not run)
```

Description

This tool creates a virtual raster or downloads data for an extent from Cloud Optimized GeoTIFFs (COGs) from the Soil Landscapes of the United States 100-meter (SOLUS100) soil property maps project repository.

Usage

```
fetchSOLUS(
  x = NULL,
  depth_slices = c(0, 5, 15, 30, 60, 100, 150),
 variables = c("anylithicdpt", "caco3", "cec7", "claytotal", "dbovendry", "ec", "ecec",
   "fragvol", "gypsum", "ph1to1h2o", "resdept", "sandco", "sandfine", "sandmed",
   "sandtotal", "sandvc", "sandvf", "sar", "silttotal", "soc"),
  output_type = c("prediction", "relative prediction interval",
   "95% low prediction interval", "95% high prediction interval"),
  grid = TRUE,
  samples = NULL,
 method = c("linear", "constant", "fmm", "natural", "monoH.FC", "step", "slice"),
  max_depth = 151,
  filename = NULL,
  overwrite = FALSE
)
```

Arguments

x	An R spatial object (such as a *SpatVector*, *SpatRaster*, or *sf* object) or a *SoilProfileCollection* with coordinates initialized via `aqp::initSpatial<-`. Default: `NULL` returns the CONUS extent as virtual raster. If `x` is a *SpatRaster* the coordinate reference system, extent, and resolution are used as a template for the output raster.
depth_slices	character. One or more of: `"0"`, `"5"`, `"15"`, `"30"`, `"60"`, `"100"`, `"150"`. The "depth slice" `"all"` (used for variables such as `"anylithicdpt"`, and `"resdept"`) is always included if any site-level variables are selected.
variables	character. One or more of: `"anylithicdpt"`, `"caco3"`, `"cec7"`, `"claytotal"`, `"dbovendry"`, `"ec"`, `"ecec"`, `"fragvol"`, `"gypsum"`, `"ph1to1h2o"`, `"resdept"`, `"sandco"`, `"sandfine"`, `"sandmed"`, `"sandtotal"`, `"sandvc"`, `"sandvf"`, `"sar"`, `"silttotal"`, `"soc"`.
output_type	character. One or more of: `"prediction"`, `"relative prediction interval"`, `"95% low prediction interval"`, `"95% high prediction interval"`
grid	logical. Default `TRUE` returns a *SpatRaster* object for an extent. `FALSE` returns a *SoilProfileCollection*. Any other value returns a *list* object with names `"grid"` and `"spc"` containing both result objects.
samples	integer. Number of regular samples to return for *SoilProfileCollection* output. Default `NULL` will convert all grid cells to a unique profile. Note that for a large extent, this can produce large objects with a very large number of layers (especially with `method` other than `"step"`).

method	character. Used to determine depth interpolation method for *SoilProfileCollection* output. Default: "linear". Options include any method allowed for approxfun() or splinefun() plus "step" and "slice". "step" uses the prediction depths as the top and bottom of each interval to create a piecewise continuous profile to maximum of 200 cm depth (for 150 cm upper prediction depth). "slice" returns a discontinuous profile with 1 cm thick slices at the predicted depths. Both "step" and "slice" return a number of layers equal to length of depth_slices, and all other methods return data in interpolated 1cm slices.
max_depth	integer. Maximum depth to interpolate 150 cm slice data to. Default: 151. Interpolation deeper than 151 cm is not possible for methods other than "step" and will result in missing values.
filename	character. Path to write output raster file. Default: NULL will keep result in memory (or store in temporary file if memory threshold is exceeded)
overwrite	Overwrite filename if it exists? Default: FALSE

Details

If the input object x is not specified (NULL or missing), a *SpatRaster* object using the virtual URLs is returned. The full extent and resolution data set can be then downloaded and written to file using terra::writeRaster() (or any other processing step specifying an output file name). When input object x is specified, a *SpatRaster* object using in memory or local (temporary file or filename) resources is returned after downloading the data only for the target extent. In the case where x is a *SoilProfileCollection* or an *sf* or *SpatVector* object containing point geometries, the result will be a *SoilProfileCollection* for values extracted at the point locations. To return both the *SpatRaster* and *SoilProfileCollection* object output in a *list*, use grid = NULL.

Value

A *SpatRaster* object containing SOLUS grids for specified extent, depths, variables, and product types.

Author(s)

Andrew G. Brown

References

Nauman, T. W., Kienast-Brown, S., Roecker, S. M., Brungard, C., White, D., Philippe, J., & Thompson, J. A. (2024). Soil landscapes of the United States (SOLUS): developing predictive soil property maps of the conterminous United States using hybrid training sets. Soil Science Society of America Journal, 88, 2046–2065. doi:10.1002/saj2.20769

Examples

```
## Not run:
b <- c(-119.747629, -119.67935, 36.912019, 36.944987)

bbox.sp <- sf::st_as_sf(wk::rct(
```

```
  xmin = b[1], xmax = b[2], ymin = b[3], ymax = b[4],
  crs = sf::st_crs(4326)
))

ssurgo.geom <- soilDB::SDA_spatialQuery(
  bbox.sp,
  what = 'mupolygon',
  db = 'SSURGO',
  geomIntersection = TRUE
)

# grid output
res <- fetchSOLUS(
  ssurgo.geom,
  depth_slices = "0",
  variables = c("sandtotal", "silttotal", "claytotal", "cec7"),
  output_type = "prediction"
)

terra::plot(res)

# SoilProfileCollection output, using linear interpolation for 1cm slices
# site-level variables (e.g. resdept) added to site data.frame of SPC
res <- fetchSOLUS(
  ssurgo.geom,
  depth_slices = c("0", "5", "15", "30", "60", "100", "150"),
  variables = c("sandtotal", "silttotal", "claytotal", "cec7", "resdept"),
  output_type = "prediction",
  method = "linear",
  grid = FALSE,
  samples = 10
)

# plot, truncating each profile to the predicted restriction depth
aqp::plotSPC(trunc(res, 0, res$resdept_p), color = "claytotal_p", divide.hz = FALSE)

## End(Not run)
```

`fetchSRI` *Fetch Soil Inventory Resource (SRI) for USFS Region 6*

Description

This is a higher level wrapper around the get_SRI and get_SRI_layers functions. This function can fetch multiple File Geodatabases (GDB) and returns all the layers within the GDB.

Usage

```
fetchSRI(gdb, ...)
```

Arguments

gdb	A `character` vector of the GDB(s), e.g. `'Deschutes'`.
...	Arguments to pass to get_SRI.

Value

A list.

Author(s)

Josh Erickson

See Also

`get_SRI()` `get_SRI_layers()`

Examples

```
## Not run:

# fetch Willamette and Winema SRI

sri <- fetchSRI(gdb = c('will', 'win'), quiet = TRUE)


## End(Not run)
```

---

fetchVegdata *Get vegetation plot data from local NASIS database*

Description

Convenience function for loading most commonly used Vegetation Plot information from local NASIS database.

Usage

```
fetchVegdata(
  SS = TRUE,
  include_pedon = TRUE,
  nullFragsAreZero = TRUE,
  stringsAsFactors = NULL,
  dsn = NULL
)

get_vegplot_from_NASIS_db(SS = TRUE, stringsAsFactors = NULL, dsn = NULL)

get_vegplot_location_from_NASIS_db(
```

```
  SS = TRUE,
  stringsAsFactors = NULL,
  dsn = NULL
)

get_vegplot_trhi_from_NASIS_db(SS = TRUE, stringsAsFactors = NULL, dsn = NULL)

get_vegplot_species_from_NASIS_db(
  SS = TRUE,
  stringsAsFactors = NULL,
  dsn = NULL
)

get_vegplot_transect_from_NASIS_db(
  SS = TRUE,
  stringsAsFactors = NULL,
  dsn = NULL
)

get_vegplot_transpecies_from_NASIS_db(
  SS = TRUE,
  stringsAsFactors = NULL,
  dsn = NULL
)

get_vegplot_transpoints_from_NASIS_db(SS = TRUE, dsn = NULL)

get_vegplot_prodquadrats_from_NASIS_db(SS = TRUE, dsn = NULL)

get_vegplot_groundsurface_from_NASIS_db(SS = TRUE, dsn = NULL)

get_vegplot_tree_si_summary_from_NASIS_db(
  SS = TRUE,
  stringsAsFactors = NULL,
  dsn = NULL
)

get_vegplot_speciesbasalarea_from_NASIS(SS = TRUE, dsn = NULL)

get_vegplot_tree_si_details_from_NASIS_db(
  SS = TRUE,
  stringsAsFactors = NULL,
  dsn = NULL
)

get_vegplot_textnote_from_NASIS_db(
  SS = TRUE,
  fixLineEndings = TRUE,
```

```
  stringsAsFactors = NULL,
  dsn = NULL
)
```

Arguments

SS	fetch data from the currently loaded selected set in NASIS or from the entire local database (default: TRUE)
include_pedon	Include pedon and transect data joined to site? (default: TRUE). If include_pedon is set to "assocuserpedonid" only pedon records that are linked through the Associated User Pedon ID column will have their peiid reported in the vegplot table.
nullFragsAreZero	Should fragment volumes of NULL be interpreted as 0? (default: TRUE), see details
stringsAsFactors	deprecated
dsn	Optional: path to local SQLite database containing NASIS table structure; default: NULL
fixLineEndings	Replace '\r\n' with '\n'; Default: TRUE

Value

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

get_vegplot_location_from_NASIS_db(): a data.frame containing location data from the corresponding record in the site table

get_vegplot_trhi_from_NASIS_db(): a data.frame containing Rangeland Health Indicator (RHI) data from the vegplot table

get_vegplot_species_from_NASIS_db(): a data.frame containing Plot Plant Inventory data

get_vegplot_transect_from_NASIS_db(): a data.frame containing Vegetation Transect data

get_vegplot_transect_from_NASIS_db(): a data.frame containing Vegetation Transect Plant Summary data

get_vegplot_transpoints_from_NASIS_db(): a data.frame containing Vegetation Transect Point Plant Cover Details

get_vegplot_prodquadrats_from_NASIS_db(): a data.frame containing Vegetation Transect Production Quadrat data

get_vegplot_groundsurface_from_NASIS_db(): a data.frame containing summary data for line point intercept ground surface cover hits by cover type.

get_vegplot_tree_si_summary_from_NASIS_db(): a data.frame containing Vegetation Plot Tree Site Index Summary data

get_vegplot_speciesbasalarea_from_NASIS(): a data.frame containing Vegetation Plot Species Basal Area and Trees Counted data

`get_vegplot_tree_si_details_from_NASIS_db()`: a data.frame containing Vegetation Plot Tree Site Index Details data

`get_vegplot_textnote_from_NASIS_db()`: a data.frame containing Vegetation Plot text notes

Examples

```
vsurf <- get_vegplot_groundsurface_from_NASIS_db()
```

`filter_geochem` *Filter KSSL Geochemical Table*

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, subset 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

```
format_SQL_in_statement(c(2648889L, 2648890L))
```

getHzErrorsNASIS *Get Logic Errors in NASIS/PedonPC Pedon Horizon*

Description

Get Logic Errors in NASIS/PedonPC Pedon Horizon

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; default: NULL

Value

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

---

get_colors_from_NASIS_db

Get 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,
  method = "dominant",
  mixColors = FALSE,
  dsn = NULL
)
```

Arguments

SS	fetch data from Selected Set in NASIS or from the entire local database (default: TRUE)
method	Aggregation method to handle multiple colors per horizon and moisture state. Default "dominant" for dominant condition (or first record) within moisture state. Other options include "mixed" to calculate mixture using simplifyColorData() and "none" to do no aggregation (returns a long format representation that may have multiple values per horizon and moisture state)
mixColors	Deprecated. See method. Should mixed colors be calculated where multiple colors are populated for the same moisture state in a horizon? Default FALSE takes the dominant color based on colorpct or first record based on horizon ID (phiid) sorting for "moist" and "dry" state. Pedon Horizon Color records without a moisture state populated are ignored.
dsn	Optional: path to local SQLite database containing NASIS table structure; default: NULL

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

Get 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.

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` *Get component month data from a local NASIS Database*

Description

Get component month data from a local NASIS Database.

Usage

```
get_comonth_from_NASIS_db(
  SS = TRUE,
  fill = FALSE,
  stringsAsFactors = NULL,
  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`	deprecated
`dsn`	Optional: path to local SQLite database containing NASIS table structure; default: `NULL`

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` *Get component data from a local NASIS Database*

Description

Get component data from a local NASIS Database

Usage

```
get_component_data_from_NASIS_db(
  SS = TRUE,
  nullFragsAreZero = TRUE,
  stringsAsFactors = NULL,
  dsn = NULL
)

get_component_diaghz_from_NASIS_db(SS = TRUE, dsn = NULL)

get_component_restrictions_from_NASIS_db(SS = TRUE, dsn = NULL)

get_component_correlation_data_from_NASIS_db(
  SS = TRUE,
  dropAdditional = TRUE,
  dropNotRepresentative = TRUE,
  stringsAsFactors = NULL,
  dsn = NULL
)

get_component_cogeomorph_data_from_NASIS_db(SS = TRUE, dsn = NULL)

get_component_cogeomorph_data_from_NASIS_db2(SS = TRUE, dsn = NULL)

get_component_copm_data_from_NASIS_db(
  SS = TRUE,
  stringsAsFactors = NULL,
  dsn = NULL
)

get_component_esd_data_from_NASIS_db(
  SS = TRUE,
  stringsAsFactors = NULL,
```

```
  dsn = NULL
)

get_component_otherveg_data_from_NASIS_db(SS = TRUE, dsn = NULL)

get_copedon_from_NASIS_db(SS = TRUE, dsn = NULL)

get_component_horizon_data_from_NASIS_db(
  SS = TRUE,
  fill = FALSE,
  dsn = NULL,
  nullFragsAreZero = TRUE
)
```

Arguments

SS	fetch data from the currently loaded selected set in NASIS or from the entire local database (default: TRUE)
nullFragsAreZero	should surface fragment cover percentages of NULL be interpreted as 0? (default: TRUE)
stringsAsFactors	deprecated
dsn	Optional: path to local SQLite database containing NASIS table structure; default: NULL
dropAdditional	Remove map units with "additional" status? Default: TRUE
dropNotRepresentative	Remove non-representative data map units? Default: TRUE
fill	Return a single minimal (NA-filled) horizon for components with no horizon records? Default FALSE

Value

a data.frame

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_component_from_GDB` *Get a SoilProfileCollection from a SSURGO file geodatabase*

Description

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

Usage

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

get_legend_from_GDB(
  dsn = "gNATSGO_CONUS.gdb",
  WHERE = NULL,
  droplevels = TRUE,
  stringsAsFactors = NULL,
  stats = FALSE
)

get_mapunit_from_GDB(
  dsn = "gNATSGO_CONUS.gdb",
  WHERE = NULL,
  droplevels = TRUE,
  stringsAsFactors = NULL,
  stats = FALSE
)

fetchGDB(
  dsn = "gNATSGO_CONUS.gdb",
  WHERE = NULL,
  childs = FALSE,
```

```
  droplevels = TRUE,
  stringsAsFactors = NULL
)
```

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	deprecated
stats	Return statistics (number of mapunit keys per legend; number of components, major components per mapunit, total and hydric component percentage)? Default: 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() which currently can only target one table (e.g. legend, mapunit or component) at a time 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://nrcs.app.box.com/v/soils/folder/191790828371
##
# dsn <- "D:/geodata/soils/gNATSGO_CONUS.gdb"
# le <- get_legend_from_GDB(dsn = dsn, WHERE = "areasymbol 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 = "areasymbol LIKE 'IN%'")
```

---

get_component_from_SDA *Get SSURGO/STATSGO2 Mapunit 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

```
get_component_from_SDA(
  WHERE = NULL,
  duplicates = FALSE,
  childs = TRUE,
  droplevels = TRUE,
  nullFragsAreZero = TRUE,
  stringsAsFactors = NULL
)

get_cointerp_from_SDA(
  WHERE = NULL,
  mrulename = NULL,
  duplicates = FALSE,
  droplevels = TRUE,
  stringsAsFactors = NULL
)

get_legend_from_SDA(WHERE = NULL, droplevels = TRUE, stringsAsFactors = NULL)

get_lmuaoverlap_from_SDA(
  WHERE = NULL,
  droplevels = TRUE,
  stringsAsFactors = NULL
)

get_mapunit_from_SDA(WHERE = NULL, droplevels = TRUE, stringsAsFactors = NULL)

get_chorizon_from_SDA(
```

```
  WHERE = NULL,
  duplicates = FALSE,
  childs = TRUE,
  nullFragsAreZero = TRUE,
  droplevels = TRUE,
  stringsAsFactors = NULL
)

fetchSDA(
  WHERE = NULL,
  duplicates = FALSE,
  childs = TRUE,
  nullFragsAreZero = TRUE,
  rmHzErrors = FALSE,
  droplevels = TRUE,
  stringsAsFactors = NULL
)

get_cosoilmoist_from_SDA(
  WHERE = NULL,
  duplicates = FALSE,
  impute = TRUE,
  stringsAsFactors = NULL
)
```

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
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.
nullFragsAreZero	should fragment volumes of NULL be interpreted as 0? (default: TRUE), see details
stringsAsFactors	deprecated
mrulename	character. Interpretation rule names
rmHzErrors	should pedons with horizonation errors be removed from the results? (default: FALSE)
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 alsoNULL (default: TRUE)

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 = "areasymbol = '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 = "areareasymbol != '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 `nullFragsAreZero` will have a significant impact on the rock fragment fractions returned by `fetchSDA`. Set `nullFragsAreZero = FALSE` in those cases where there are many data-gaps and NULL rock fragment values should be interpreted as NULLs. Set `nullFragsAreZero = TRUE` in those cases where NULL rock fragment values should be interpreted as 0.

Additional examples can be found in the Soil Data Access (SDA) Tutorial

Value

A data.frame or SoilProfileCollection object.

Author(s)

Stephen Roecker

See Also

SDA_query

`get_cosoilmoist_from_NASIS`

Get 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 = NULL,
  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	deprecated
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.

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_ecosite_history_from_NASIS_db

Get Site Ecological Site History

Description

Gets the Site Ecological Site History data from local NASIS database. Used by `get_extended_data_from_NASIS_db()`.

Usage

```
get_ecosite_history_from_NASIS_db(
  best = TRUE,
  SS = TRUE,
  es_classifier = NULL,
  dsn = NULL
)
```

Arguments

best	Should the "best" ecological site correlation be chosen? Creates field called `es_selection_method` with `"most recent"` or `"least missing data"` for resolving many:1 relationships in site history.
SS	Use selected set? Default: `TRUE`
es_classifier	Optional: character. Vector of classifier names (and corresponding records) to retain in final result.
dsn	Path to SQLite data source, or a `DBIConnection` to database with NASIS schema.

Value

a `data.frame`, or `NULL` on error

See Also

`get_extended_data_from_NASIS_db()`

get_EDIT_ecoclass_by_geoUnit

Get Ecological Dynamics Information Tool (EDIT) ecological sites by catalog (ESD/ESG) and MLRA

Description

Data are accessed via Ecological Dynamics Interpretive Tool (EDIT) web services: https://edit.jornada.nmsu.edu/resources/es
`geoUnit` refers to MLRA codes, possibly with a leading zero and trailing "X" for two digit MLRA symbols.

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. NULL if no result.

Examples

```
## Not run:
   get_EDIT_ecoclass_by_geoUnit(c("018X","022A"))

## End(Not run)
```

---

get_extended_data_from_NASIS_db

Get accessory tables and summaries from a local NASIS Database

---

Description

Get accessory tables and summaries from a local NASIS Database

Usage

```
get_extended_data_from_NASIS_db(
  SS = TRUE,
  nullFragsAreZero = TRUE,
  stringsAsFactors = NULL,
  dsn = NULL
)
```

Arguments

SS	get data from the currently loaded Selected Set in NASIS or from the entire local database (default: TRUE)
nullFragsAreZero	should fragment volumes of NULL be interpreted as 0? (default: TRUE), see details
stringsAsFactors	deprecated
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

Get accessory tables and summaries from a local pedonPC Database

---

Description

Get 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.

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 *Get 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,
  fill = FALSE,
  stringsAsFactors = NULL,
  dsn = NULL
)
```

Arguments

SS	fetch data from Selected Set in NASIS or from the entire local database (default: TRUE)
fill	include pedons without horizon data in result? default: FALSE
stringsAsFactors	deprecated
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

Get 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.

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

Get 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

Get 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

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_mapunit_from_NASIS

Get Legend, Mapunit and Legend Mapunit Area Overlap Tables

Description

Get Legend, Mapunit and Legend Mapunit Area Overlap Tables

Usage

```
get_mapunit_from_NASIS(
  SS = TRUE,
  repdmu = TRUE,
  droplevels = TRUE,
  stringsAsFactors = NULL,
  areatypename = c("Non-MLRA Soil Survey Area", "MLRA Soil Survey Area"),
  dsn = NULL
)

get_legend_from_NASIS(
  SS = TRUE,
  droplevels = TRUE,
  stringsAsFactors = NULL,
  areatypename = c("Non-MLRA Soil Survey Area", "MLRA Soil Survey Area"),
  dsn = NULL
)

get_lmuaoverlap_from_NASIS(
  SS = TRUE,
  droplevels = TRUE,
  stringsAsFactors = NULL,
  areatypename = c("Non-MLRA Soil Survey Area", "MLRA Soil Survey Area"),
  dsn = NULL
)
```

Arguments

SS	Fetch data from the currently loaded selected set in NASIS or from the entire local database (default: TRUE)
repdmu	Return only "representative" data mapunits? Default: TRUE
droplevels	Drop unused levels from farmlndcl and other factor levels from NASIS domains?
stringsAsFactors	deprecated
areatypename	Used for get_legend_from_NASIS(). Default: c('Non-MLRA Soil Survey Area', 'MLRA Soil Survey Area')
dsn	Optional: path to local SQLite database containing NASIS table structure; default: NULL

---

get_NASIS_metadata *Get NASIS Metadata (Domain, Column and Choice Lists)*

Description

Retrieve a table containing domain and column names with choice list labels/names/sequences/values from the NASIS 7 metadata tables.

Usage

```
get_NASIS_metadata(dsn = NULL, include_description = FALSE)

get_NASIS_column_metadata(
  x,
  what = "ColumnPhysicalName",
  include_description = FALSE,
  dsn = NULL
)
```

Arguments

dsn	Optional: path or *DBIConnection* to local database containing NASIS table structure; default: NULL
include_description	Include "ChoiceDescription" column? Default: FALSE
x	character vector to match in NASIS metadata
what	Column to match x against. Default "ColumnPhysicalName"; alternate options include "DomainID", "DomainName", "DomainRanked", "DisplayLabel", "ChoiceSequence", "ChoiceValue", "ChoiceName", "ChoiceLabel", "ChoiceObsolete", "ChoiceDescription", "ColumnLogicalName"

Details

These data are derived from the MetadataDomainDetail, MetadataDomainMaster, and MetadataTableColumn tables and help with mapping between values stored in the NASIS database and human-readable values. The human-readable values align with the values returned in public facing interfaces such as SSURGO via Soil Data Access and NASIS Web Reports. The data in these tables can also be used to create *ordered* factors where options for levels of a particular data element follow a logical `ChoiceSequence`.

If a local NASIS instance is set up, and this is the first time `get_NASIS_metadata()` has been called, the metadata will be obtained from the NASIS local database. Subsequent runs in the same session will use a copy of the data object `NASIS.metadata` cached in `soilDB.env` which can be accessed with `get_soilDB_env()$NASIS.metadata`.

For users without a local NASIS instance, a cached copy of the NASIS metadata are used (`data/metadata.rda`).

See `?soilDB::metadata` for additional details.

Value

a `data.frame` containing DomainID, DomainName, DomainRanked, DisplayLabel, ChoiceSequence, ChoiceValue, ChoiceName, ChoiceLabel, ChoiceObsolete, ColumnPhysicalName, ColumnLogicalName and optionally ChoiceDescription when `include_description=TRUE`.

a `data.frame` containing selected NASIS metadata sorted first on `DomainID` and then on `ChoiceSequence`

Examples

```
get_NASIS_column_metadata("texcl")
```

---

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

`tables`	character vector of table names
`keycol`	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_metadata`

Get NASIS Table Metadata (Table and Column Descriptions)

Description

Retrieve a table containing table and column names with descriptions, help text, units of measure, etc. from NASIS 7 metadata tables.

Usage

```
get_NASIS_table_metadata(
  table = NULL,
  column = NULL,
  what.table = "TablePhysicalName",
  what.column = "ColumnPhysicalName",
  query_string = FALSE,
  dsn = NULL
)
```

Arguments

`table`	Character vector of table identifiers to match. Default `NULL` for "all tables" (no constraint)
`column`	Character vector of column identifiers to match. Default `NULL` for "all columns" (in selected tables, if any, otherwise no constraint)
`what.table`	Column to match `table` against. Default: `TablePhysicalName`.
`what.column`	Column to match `column` against. Default: `ColumnPhysicalName`.
`query_string`	Default: `FALSE`; if `TRUE` return a character containing query that would be sent to NASIS.
`dsn`	Optional: path or *DBIConnection* to local database containing NASIS table structure; default: `NULL`

Details

These data are derived from the MetadataTable and MetadataTableColumn tables and describe the expected contents of standard NASIS tables and columns.

For NASIS choice lists based on domain and column names see `get_NASIS_metadata()` and `NASISChoiceList()`. This function (`get_NASIS_table_metadata()`) is intended for higher-level description of the expected contents of a NASIS database instance, rather than the codes/specific values used within columns.

Value

a `data.frame`

See Also

`get_NASIS_metadata()` `NASISChoiceList()` `uncode()` `code()`

Examples

```
if (local_NASIS_defined())
 str(get_NASIS_table_metadata())
```

---

get_NASIS_table_name_by_purpose

Get NASIS 7 Physical Table Names

Description

Method generalizing concepts of NASIS 7 data model to group tables by "purpose." 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 a data source.

Usage

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

Arguments

purpose	character. One or more of: "metadata", "lookup", "nasis", "site", "pedon", "transect", "component", "vegetation", "project", "techsoilservice", "area", "soilseries", "legend", "mapunit", "datamapunit"
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)
```

---

get_NOAA_GHCND *Get Global Historical Climatology Network Daily (GHCND) data from NOAA API*

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*year*datatypeid 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 (https://www.ncdc.noaa.gov/cdo-web/token)

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_NOAA_stations_nearXY`

Get NOAA station data near a given latitude and longitude

Description

Query the NOAA API to get station data (limit 1000 records) near a point. Default extent is plus or minus 0.5 degrees (bounding box) (with `bbox = 1`) around the specified point [lat, lng].

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_stations_nearXY(lat, lng, apitoken, bbox = 1, crs = "EPSG:4326")
```

Arguments

`lat`	Latitude or Y coordinate in `crs`
`lng`	Longitude or X coordinate in `crs`
`apitoken`	API key token for NOAA NCDC web service
`bbox`	Optional: Dimension of the bounding box centered at `lat`, `lng`.
`crs`	Coordinate Reference System. Default `"EPSG:4326"`

Value

data.frame containing station information for all stations within a bounding box around `lat`, `lng`.

Examples

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

# stations <- get_NOAA_stations_nearXY(lat = 37, lng = -120,
#                                      apitoken = "yourtokenhere")
```

get_OSD *Get Official Series Description Data from JSON, HTML or TXT sources*

Description

Get Official Series Description Data from JSON, HTML or TXT sources

Usage

```
get_OSD(
  series,
  base_url = NULL,
  result = c("json", "html", "txt"),
  fix_ocr_errors = FALSE,
  verbose = FALSE
)

get_OSD_JSON(series, base_url = NULL)
```

Arguments

series	A character vector of Official Series names e.g. "Chewacla"
base_url	Optional: alternate JSON/HTML/TXT repository path. Default: NULL uses "https://github.com/ncss-tech/SoilKnowledgeBase" for result="json"
result	Select "json", "html", or "txt" output
fix_ocr_errors	Default: FALSE; Applies only to result='json'. Convert clear cases of Optical Character Recognition (OCR) errors to likely actual values.
verbose	Print errors and warning messages related to HTTP requests? Default: FALSE

Details

The default base_url for result="json" is to 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-t for JSON. And "https://soilseriesdesc.sc.egov.usda.gov/OSD_Docs/{LETTER}/{SERIES}.html is for result="html" (official source).

fix_ocr_errors by default is turned off (FALSE). When TRUE, assume that in color data hue/value/chroma lowercase "L" ("l") is a 1, and a capital "O" is interpreted as zero. Also, in horizon designations assume lowercase "L" is a 1, and a string that starts with 0 starts with the capital letter "O".

Value

For JSON result: A data.frame with 1 row per series, and 1 column per "section" in the OSD as defined in National Soil Survey Handbook. For TXT or HTML result a list of character vectors containing OSD text with 1 element per series and one value per line.

Examples

```
series <- c("Musick", "Hector", "Chewacla")
get_OSD(series)
```

`get_phroots_from_NASIS_db`

Get pedon horizon roots data from a local NASIS Database

Description

This function returns records from the `phroots` table of a local NASIS database. Pedon and pedon horizon record IDs are also included for linking back to related records, typically queried via `fetchNASIS()`.

Usage

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

Arguments

`SS`	logical, limit query to the selected set
`dsn`	optional path or *DBIConnection* to local database containing NASIS table structure; default: `NULL`

Value

A `data.frame`

`get_projectmilestone_from_NASIS`

Get NASIS Project Information

Description

Helper functions for accessing the NASIS Project object.

Usage

```
get_projectmilestone_from_NASIS(SS = TRUE, stringsAsFactors = NULL, dsn = NULL)

get_projectmapunit_from_NASIS(SS = TRUE, stringsAsFactors = NULL, dsn = NULL)
```

Arguments

SS	*logical*. Use selected set? Default: TRUE. Set FALSE for local database.
stringsAsFactors	Deprecated.
dsn	Optional: path or *DBIConnection* to local database containing NASIS table structure; default: NULL

Value

get_projectmilestone_from_NASIS(): *data.frame* containing project and project milestone information

get_projectmapunit_from_NASIS(): *data.frame* containing project and project mapunit information

---

get_RMF_from_NASIS_db *Get RMF data from local NASIS*

Description

Prepare a list of data.frame objects with data from the "phrdxfeatures" and "phredoxfcolor" tables. These tables are related by "phrdxfiid" column, and related to horizon data via "phiid".

Usage

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

Arguments

SS	logical, limit query to the selected set
dsn	optional path or *DBIConnection* to local database containing NASIS table structure; default: NULL

Value

a list with two data.frame objects:

- RMF: contents of "phrdxfeatures" table, often >1 row per horizon
- RMF_colors: contents of "phredoxfcolor", usually >1 row per record in "phrdxfeatures"

get_SDA_coecoclass *Get mapunit ecological sites from Soil Data Access*

Description

Get mapunit ecological sites from Soil Data Access

Usage

```
get_SDA_coecoclass(
  method = "None",
  areasymbols = NULL,
  mukeys = NULL,
  WHERE = NULL,
  query_string = FALSE,
  ecoclasstypename = c("NRCS Rangeland Site", "NRCS Forestland Site"),
  ecoclassref = "Ecological Site Description Database",
  not_rated_value = "Not assigned",
  miscellaneous_areas = TRUE,
  include_minors = TRUE,
  threshold = 0,
  dsn = NULL
)
```

Arguments

method	aggregation method. One of: "Dominant Component", "Dominant Condition", "All" or "None" (default). If method="all" multiple numbered columns represent site composition within each map unit e.g. site1..., site2.... If method="none" is selected one row will be returned per *component*; in all other cases one row will be returned per *map unit*.
areasymbols	vector of soil survey area symbols
mukeys	vector of map unit keys
WHERE	character containing SQL WHERE clause specified in terms of fields in legend, mapunit, component or coecosite tables, used in lieu of mukeys or areasymbols
query_string	Default: FALSE; if TRUE return a character string containing query that would be sent to SDA via SDA_query
ecoclasstypename	Default: c("NRCS Rangeland Site", "NRCS Forestland Site"). If NULL no constraint on ecoclasstypename is used in the query.
ecoclassref	Default: "Ecological Site Description Database". If NULL no constraint on ecoclassref is used in the query.
not_rated_value	Default: "Not assigned"

`miscellaneous_areas`
: logical. Include miscellaneous areas (non-soil components)?

`include_minors`
: logical. Include minor components? Default: `TRUE`.

`threshold`
: integer. Default: `0`. Minimum combined component percentage (RV) for inclusion of a mapunit's ecological site in wide-format tabular summary. Used only for `method="all"`.

`dsn`
: Path to local SQLite database or a DBIConnection object. If `NULL` (default) use Soil Data Access API via `SDA_query()`.

Details

When `method="Dominant Condition"` an additional field `ecoclasspct_r` is returned in the result with the sum of `comppct_r` that have the dominant condition `ecoclassid`. The component with the greatest `comppct_r` is returned for the `component` and `coecosite` level information.

Note that if there are multiple `coecoclasskey` per `ecoclassid` there may be more than one record per component.

---

get_SDA_cosurfmorph *Get Geomorphic/Surface Morphometry Data from Soil Data Access*

Description

Get Geomorphic/Surface Morphometry Data from Soil Data Access or a local SSURGO data source and summarize by counts and proportions ("probabilities").

Usage

```
get_SDA_cosurfmorph(
  table = c("cosurfmorphgc", "cosurfmorphhpp", "cosurfmorphss", "cosurfmorphmr"),
  by = "mapunit.mukey",
  areasymbols = NULL,
  mukeys = NULL,
  WHERE = NULL,
  method = c("bygroup", "none"),
  include_minors = TRUE,
  miscellaneous_areas = FALSE,
  representative_only = TRUE,
  db = c("SSURGO", "STATSGO"),
  dsn = NULL,
  query_string = FALSE
)
```

Arguments

table	Target table to summarize. Default: "cosurfmorphgc" (3D Geomorphic Component). Alternate choices include cosurfmorphhpp (2D Hillslope Position), cosurfmorphss (Surface Shape), and cosurfmorphmr (Microrelief).
by	Grouping variable. Default: "mapunit.mukey"
areasymbols	A vector of soil survey area symbols (e.g. 'CA067')
mukeys	A vector of map unit keys (e.g. 466627)
WHERE	WHERE clause added to SQL query. For example: areasymbol = 'CA067'
method	*character*. One of: "ByGroup", "None"
include_minors	logical. Include minor components? Default: TRUE.
miscellaneous_areas	logical. Include miscellaneous areas (non-soil components) in results? Default: FALSE.
representative_only	logical. Include only representative Component Parent Material Groups? Default: TRUE.
db	Either 'SSURGO' (default) or 'STATSGO'. If 'SSURGO' is specified areasymbol = 'US' records are excluded. If 'STATSGO' only areasymbol = 'US' records are included.
dsn	Path to local SSURGO database SQLite database. Default NULL uses Soil Data Access.
query_string	Return query instead of sending to Soil Data Access / local database. Default: FALSE.

Details

Default table="cosurfmorphgc" summarizes columns geomposmntn, geomposhill, geomposflats, and geompostrce. table="cosurfmorphhpp" summarizes "hillslopeprof", table="cosurfmorphss" summarizes shapeacross and shapedown, and table="cosurfmorphmr" summarizes geomicrorelief.

Queries are a generalization of now-deprecated functions from sharpshootR package by Dylan Beaudette: geomPosMountainProbability(), geomPosHillProbability(), surfaceShapeProbability(), hillslopeProbability()

Similar summaries of SSURGO component surface morphometry data by series name can be found in fetchOSD(, extended=TRUE) or downloaded from https://github.com/ncss-tech/SoilWeb-data Full component data including surface morphometry summaries at the "site" level can be obtained with fetchSDA().

Value

a data.frame containing the grouping variable (by) and tabular summaries of counts and proportions of geomorphic records.

Author(s)

Dylan E. Beaudette, Andrew G. Brown

See Also

fetchSDA() get_SDA_pmgroupname()

Examples

```
## Not run:
 # Summarize by 3D geomorphic components by component name (default `by='compname'`)
 get_SDA_cosurfmorph(WHERE = "areasymbol = 'CA630'")

 # Whole Soil Survey Area summary (using `by = 'areasymbol'`)
 get_SDA_cosurfmorph(by = 'areasymbol', WHERE = "areasymbol = 'CA630'")

 # 2D Hillslope Position summary(using `table = 'cosurfmorphhpp'`)
 get_SDA_cosurfmorph('cosurfmorphhpp', WHERE = "areasymbol = 'CA630'")

 # Surface Shape summary (using `table = 'cosurfmorphss'`)
 get_SDA_cosurfmorph('cosurfmorphss', WHERE = "areasymbol = 'CA630'")

 # Microrelief summary (using `table = 'cosurfmorphmr'`)
 get_SDA_cosurfmorph('cosurfmorphmr', WHERE = "areasymbol = 'CA630'")

## End(Not run)
```

---

get_SDA_hydric *Get map unit hydric soils information from Soil Data Access*

Description

Assess the hydric soils composition of a map unit.

Usage

```
get_SDA_hydric(
  areasymbols = NULL,
  mukeys = NULL,
  WHERE = NULL,
  method = "MAPUNIT",
  include_minors = TRUE,
  miscellaneous_areas = TRUE,
  query_string = FALSE,
  dsn = NULL
)
```

Arguments

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

WHERE	character containing SQL WHERE clause specified in terms of fields in legend, mapunit, or component tables, used in lieu of mukeys or areasymbols
method	One of: "Mapunit", "Dominant Component", "Dominant Condition", "None"
include_minors	logical. Include minor components? Default: TRUE.
miscellaneous_areas	*logical*. Include miscellaneous areas (non-soil components) in results? Default: TRUE.
query_string	Default: FALSE; if TRUE return a character string containing query that would be sent to SDA via SDA_query()
dsn	Path to local SQLite database or a DBIConnection object. If NULL (default) use Soil Data Access API via SDA_query().

Details

The default classes for method="MAPUNIT" are as follows:

- 'Nonhydric' - no hydric components
- 'Hydric' - all hydric components
- 'Predominantly Hydric' - hydric component percentage is 50% or more
- 'Partially Hydric' - one or more of the major components is hydric
- 'Predominantly Nonhydric' - hydric component percentage is less than 50%

The default result will also include the following summaries of component percentages: total_comppct, hydric_majors and hydric_inclusions.

Default method "Mapunit" produces aggregate summaries of all components in the mapunit. Use "Dominant Component" and "Dominant Condition" to get the dominant component (highest percentage) or dominant hydric condition (similar conditions aggregated across components), respectively. Use "None" for no aggregation (one record per component).

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,
  WHERE = NULL,
  include_minors = TRUE,
  miscellaneous_areas = TRUE,
  query_string = FALSE,
  not_rated_value = NA_real_,
  wide_reason = FALSE,
  dsn = NULL
)
```

Arguments

rulename	character vector of interpretation rule names (matching 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.
areasymbols	vector of soil survey area symbols
mukeys	vector of map unit keys
WHERE	character containing SQL WHERE clause specified in terms of fields in legend, mapunit, or component tables, used in lieu of mukeys or areasymbols
include_minors	logical. Include minor components? Default: TRUE.
miscellaneous_areas	*logical*. Include miscellaneous areas (non-soil components) in results? Default: TRUE.
query_string	Default: FALSE; if TRUE return a character string containing query that would be sent to SDA via SDA_query
not_rated_value	used where rating class is "Not Rated". Default: NA_real_
wide_reason	Default: FALSE; if TRUE apply post-processing to all columns with prefix "reason_" to create additional columns for sub-rule ratings.

dsn — Path to local SQLite database or a DBIConnection object. If NULL (default) use Soil Data Access API via SDA_query().

Details

Rule Names in `cointerp` table:

- AGR - Avocado Root Rot Hazard (CA)
- AGR - California Revised Storie Index (CA)
- AGR - Hops Site Suitability (WA)
- AGR - Map Unit Cropland Productivity (MN)
- AGR - Nitrate Leaching Potential, Nonirrigated (WA)
- AGR - No Till (TX)
- AGR - Pesticide Loss Potential-Soil Surface Runoff (NE)
- AGR - Ridge Till (TX)
- AGR - Selenium Leaching Potential (CO)
- AGR - Water Erosion Potential (NE)
- AGR - Wind Erosion Potential (TX)
- AGR - Winter Wheat Yield (MT)
- AGR-Pesticide and Nutrient Runoff Potential (ND)
- AGR-Rooting Depth (ND)
- American Wine Grape Varieties Site Desirability (Long)
- American Wine Grape Varieties Site Desirability (Medium)
- American Wine Grape Varieties Site Desirability (Very Long)
- AWM - Animal Mortality Disposal (Catastrophic) (MO)
- AWM - Irrigation Disposal of Wastewater (OH)
- AWM - Irrigation Disposal of Wastewater (VT)s
- AWM - Land Application of Municipal Biosolids, summer (OR)
- AWM - Manure and Food Processing Waste (MD)
- AWM - Manure and Food Processing Waste (OH)
- AWM - Overland Flow Process Treatment of Wastewater (VT)
- AWM - Rapid Infil Disposal of Wastewater (DE)
- AWM - Sensitive Soil Features (MN)
- AWM - Sensitive Soil Features (WI)
- BLM - Fencing
- BLM - Fire Damage Susceptibility
- BLM - Mechanical Treatment, Rolling Drum
- BLM - Rangeland Drill
- BLM - Rangeland Seeding, Colorado Plateau Ecoregion
- BLM - Rangeland Seeding, Great Basin Ecoregion
- BLM-Reclamation Suitability (MT)
- CLASS RULE - Depth to lithic bedrock (5 classes) (NPS)
- CLASS RULE - Soil Inorganic Carbon kg/m2 to 2m (NPS)
- CLASS RULE - Soil Organic Carbon kg/m2 to 2m (NPS)

- CLR-pastureland limitation (IN)
- Commodity Crop Productivity Index (Soybeans) (TN)
- CPI - Alfalfa Hay, NIRR - Palouse, Northern Rocky Mtns. (WA)
- CPI - Barley, IRR - Eastern Idaho Plateaus (ID)
- CPI - Grass Hay, IRR - Klamath Valleys and Basins (OR)
- CPI - Small Grains, IRR - Snake River Plains (ID)
- CPI - Wheat, IRR - Eastern Idaho Plateaus (ID)
- CZSS - Salinization due to Coastal Saltwater Inundation (CT)
- DHS - Catastrophic Event, Large Animal Mortality, Burial
- DHS - Catastrophic Mortality, Large Animal Disposal, Pit
- DHS - Catastrophic Mortality, Large Animal Disposal, Trench
- DHS - Potential for Radioactive Bioaccumulation
- DHS - Potential for Radioactive Sequestration
- DHS - Suitability for Composting Medium and Final Cover
- ENG - Construction Materials; Gravel Source
- ENG - Construction Materials; Gravel Source (AK)
- ENG - Construction Materials; Gravel Source (ID)
- ENG - Construction Materials; Gravel Source (OH)
- ENG - Construction Materials; Gravel Source (VT)
- ENG - Construction Materials; Gravel Source (WA)
- ENG - Construction Materials; Roadfill (OH)
- ENG - Construction Materials; Sand Source (OR)
- ENG - Construction Materials; Sand Source (WA)
- ENG - Construction Materials; Topsoil (GA)
- ENG - Construction Materials; Topsoil (MD)
- ENG - Daily Cover for Landfill
- ENG - Daily Cover for Landfill (AK)
- ENG - Disposal Field Suitability Class (NJ)
- ENG - Dwellings W/O Basements (OH)
- ENG - Dwellings with Basements (AK)
- ENG - Large Animal Disposal, Pit (CT)
- ENG - Lawn, landscape, golf fairway (CT)
- ENG - Lined Retention Systems
- ENG - Local Roads and Streets (OH)
- ENG - On-Site Waste Water Absorption Fields (MO)
- ENG - Septic Tank Absorption Fields
- ENG - Septic Tank Absorption Fields (MD)
- ENG - Septic Tank Absorption Fields (TX)
- ENG - Septic Tank, Gravity Disposal (TX)
- ENG - Sewage Lagoons
- ENG - Small Commercial Buildings (OH)
- ENG - Soil Potential Ratings of SSDS (CT)

- FOR (USFS) - Road Construction/Maintenance (Natural Surface)
- FOR - Compaction Potential (WA)
- FOR - Conservation Tree/Shrub Groups (MT)
- FOR - Damage by Fire (OH)
- FOR - General Harvest Season (VT)
- FOR - Hand Planting Suitability
- FOR - Hand Planting Suitability, MO13 (DE)
- FOR - Hand Planting Suitability, MO13 (MD)
- FOR - Log Landing Suitability
- FOR - Log Landing Suitability (ME)
- FOR - Log Landing Suitability (VT)
- FOR - Log Landing Suitability (WA)
- FOR - Mechanical Planting Suitability (CT)
- FOR - Mechanical Planting Suitability, MO13 (MD)
- FOR - Mechanical Site Preparation (Deep)
- FOR - Mechanical Site Preparation (Deep) (DE)
- FOR - Mechanical Site Preparation (Surface) (DE)
- FOR - Mechanical Site Preparation (Surface) (MI)
- FOR - Mechanical Site Preparation; Surface (ME)
- FOR - Potential Erosion Hazard, Road/Trail, Spring Thaw (AK)
- FOR - Potential Seedling Mortality (PIA)
- FOR - Potential Seedling Mortality(ME)
- FOR - Puddling Hazard
- FOR - Road Suitability (Natural Surface) (ME)
- FOR - Road Suitability (Natural Surface) (WA)
- FOR - Soil Rutting Hazard (OH)
- FOR - Soil Sustainability Forest Biomass Harvesting (CT)
- FOR - White Oak Suitability (MO)
- FOR-Biomass Harvest (WI)
- FOTG - Indiana Corn Yield Calculation (IN)
- GRL - Excavations to 24 inches for Plastic Pipelines (TX)
- GRL - Fencing, 24 inch Post Depth (MT)
- GRL - NV range seeding (Wind C = 100) (NV)
- GRL - NV range seeding (Wind C = 40) (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 - Rangeland Planting by Mechanical Seeding (TX)
- GRL - Rangeland Root Plowing (TX)
- Hybrid Wine Grape Varieties Site Desirability (Long)
- 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 - Trafficability Veh. Type 1 50-passes wet season (DOD)
- MIL - Trafficability Veh. Type 2 50-passes wet 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 6 50-passes wet season (DOD)
- MIL - Trafficability Veh. Type 7 50-passes wet season (DOD)
- MIL - Trafficability Veh. Type 7 dry season (DOD)
- NCCPI - Irrigated National Commodity Crop Productivity Index
- Nitrogen Loss Potential (ND)
- Potential Windthrow Hazard (TN)
- REC - Foot and ATV Trails (AK)
- REC - Playgrounds (AK)
- Reclamation Suitability (ND)
- RSK-risk assessment for manure application (OH)
- SAS - CMECS Substrate Origin
- SAS - CMECS Substrate Subclass/Group/Subgroup
- SAS - Mooring Anchor - Deadweight
- Septic System A/B Soil System (Alternate) (PA)
- Septic System CO-OP RFS III w/Spray Irrigation (PA)
- Septic System Dual Field Trench (conventional) (WV)
- Septic System Elevated Field (alternative) (WV)
- Septic System In Ground Trench (conventional) (PA)
- Septic System In Ground Trench (conventional) (WV)
- AGR - Filter Strips (TX)
- AGR - Hops Site Suitability (ID)
- AGR - Mulch Till (TX)
- AGR - Nitrate Leaching Potential, Nonirrigated (MT)
- AGR - Nitrate Leaching Potential, Nonirrigated (WV)
- AGR - No Till (VT)
- AGR - Oats Yield (MT)
- AGR - Pesticide Loss Potential-Leaching
- AGR - Pesticide Loss Potential-Leaching (NE)
- AGR - Rutting Hazard =< 10,000 Pounds per Wheel (TX)
- AGR - S. Highbush Blueberry Suitability MLRA 153 (SC)
- AGR - Wind Erosion Potential (NE)
- AGR-Available Water Capacity (ND)
- AGR-Physical Limitations (ND)
- AGR-Sodicity (ND)
- AGR-Surface Crusting (ND)
- AGR-Wind Erosion (ND)

- AWM - Irrigation Disposal of Wastewater (DE)
- AWM - Land App of Municipal Sewage Sludge (DE)
- AWM - Land App of Municipal Sewage Sludge (MD)
- AWM - Land Application of Milk (CT)
- AWM - Land Application of Municipal Biosolids, spring (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 (VT)
- AWM - Rapid Infil Disposal of Wastewater (MD)
- AWM - Rapid Infiltration Disposal of Wastewater (VT)
- AWM - Slow Rate Process Treatment of Wastewater (VT)
- BLM - Chaining Suitability
- BLM - Fugitive Dust Resistance
- BLM - Soil Restoration Potential
- BLM - Yellow Star-thistle Invasion Susceptibility
- CLASS RULE - Depth to non-lithic bedrock (5 classes) (NPS)
- CLR-cropland limitation for corn and soybeans (IN)
- Commodity Crop Productivity Index (Corn) (WI)
- CPI - Grass Hay, NIRR - Klamath Valleys and Basins (OR)
- CPI - Potatoes Productivity Index (AK)
- CPI - Potatoes, IRR - Eastern Idaho Plateaus (ID)
- CPI - Small Grains, NIRR - Palouse Prairies (ID)
- DHS - Emergency Animal Mortality Disposal by Shallow Burial
- DHS - Rubble and Debris Disposal, Large-Scale Event
- ENG - Aquifer Assessment - 7081 (MN)
- ENG - Construction Materials - Gravel Source (MN)
- ENG - Construction Materials; Gravel Source (MI)
- ENG - Construction Materials; Gravel Source (OR)
- ENG - Construction Materials; Reclamation
- ENG - Construction Materials; Reclamation (OH)
- ENG - Construction Materials; Sand Source
- ENG - Construction Materials; Sand Source (AK)
- ENG - Construction Materials; Sand Source (ID)
- ENG - Construction Materials; Sand Source (IN)
- ENG - Construction Materials; Sand Source (OH)
- ENG - Construction Materials; Topsoil
- ENG - Construction Materials; Topsoil (WA)
- ENG - Ground-based Solar Arrays, Soil-based Anchor Systems
- ENG - Local Roads and Streets

- ENG - New Ohio Septic Rating (OH)
- ENG - Sanitary Landfill (Trench) (OH)
- ENG - Septic Tank Absorption Fields (AK)
- ENG - Septic Tank Absorption Fields (DE)
- ENG - Septic Tank Absorption Fields (NY)
- ENG - Sewage Lagoons (OH)
- ENG - Shallow Excavations (AK)
- ENG - Shallow Excavations (MI)
- ENG - Unpaved Local Roads and Streets
- FOR - Black Walnut Suitability Index (MO)
- FOR - Conservation Tree and Shrub Groups (TX)
- FOR - Construction Limitations - Haul Roads/Log Landing (OH)
- FOR - Construction Limitations For Haul Roads (MI)
- FOR - Hand Planting Suitability (ME)
- FOR - Harvest Equipment Operability (MD)
- FOR - Harvest Equipment Operability (OH)
- FOR - Harvest Equipment Operability (VT)
- FOR - Mechanical Planting Suitability
- FOR - Mechanical Planting Suitability (ME)
- FOR - Mechanical Planting Suitability, MO13 (DE)
- FOR - Potential Erosion Hazard (Off-Road/Off-Trail)
- FOR - Potential Erosion Hazard (Road/Trail) (PIA)
- FOR - Potential Seedling Mortality (VT)
- FOR - Potential Windthrow Hazard (NY)
- FOR - Potential Windthrow Hazard (VT)
- FOR - Puddling Potential (WA)
- FOR - Road Suitability (Natural Surface)
- FOR - Road Suitability (Natural Surface) (OH)
- FOR - Road Suitability (Natural Surface) (OR)
- FOR - Rutting Hazard by Season
- FOR - Shortleaf pine littleleaf disease susceptibility
- FOR - Soil Compactibility Risk
- FOR - Soil Rutting Hazard (ME)
- FOR - Windthrow Hazard
- FOR-Construction Limitations for Haul Roads/Log Landings(ME)
- FOTG - Indiana Slippage Potential (IN)
- Gravity Full Depth Septic System (DE)
- GRL - Fencing, Post Depth =<36 inches
- GRL - NV range seeding (Wind C = 50) (NV)
- GRL - Ranch Access Roads (TX)
- GRL - Rangeland Roller Chopping (TX)
- Ground Penetrating Radar Penetration

- Ground-based Solar Arrays_bedrock(ME)
- Ground-based Solar Arrays_bedrock_slope_ballast(ME)
- Hybrid Wine Grape Varieties Site Desirability (Short)
- ISDH Septic Tank Interpretation (IN)
- Land Application of Municipal Sewage Sludge (PA)
- MIL - Helicopter Landing Zones (DOD)
- MIL - Trafficability Veh. Type 2 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 7 1-pass wet season (DOD)
- NCCPI - National Commodity Crop Productivity Index (Ver 3.0)
- REC - Camp and Picnic Areas (AK)
- REC - Picnic Areas (CT)
- REC - Playgrounds (CT)
- SAS - CMECS Substrate Subclass
- Septic System Drip Irrigation (Alternate) (PA)
- Septic System Free Access Sand Filter w/Drip Irrigation (PA)
- Septic System In Ground Bed (conventional) (PA)
- Septic System Peat Based Option1 (UV & At-Grade Bed)Alt (PA)
- Septic System Peat Sys Opt3 w/Subsurface Sand Filter (PA)
- Septic System Sand Mound Bed or Trench (PA)
- Septic System Shallow Placement Pressure Dosed (Alt.) (PA)
- SOH - Aggregate Stability (ND)
- SOH - Agricultural Organic Soil Subsidence
- SOH - Dynamic Soil Properties Response to Biochar
- SOH - Organic Matter Depletion
- SOIL HEALTH ASSESSMENT (NJ)
- URB - Commercial Brick Bldg; w/Reinforced Concrete Slab (TX)
- URB - Reinforced Concrete Slab (TX)
- URB/REC - Camp Areas
- URB/REC - Camp Areas (OH)
- URB/REC - Off-Road Motorcycle Trails (OH)
- URB/REC - Paths and Trails (OH)
- URB/REC - Picnic Areas
- URB/REC - Playgrounds
- URB/REC - Playgrounds (GA)
- Vinifera Wine Grape Site Desirability (Short to Medium)
- WLF - Irr. Domestic Grasses & Legumes for Food & Cover (TX)
- WLF - Upland Coniferous Trees (TX)
- WLF - Upland Deciduous Trees (TX)
- WLF - Upland Desertic Shrubs & Trees (TX)
- WLF - Upland Native Herbaceous Plants (TX)

- WLF - Upland Shrubs & Vines (TX)
- WLF-Soil Suitability - Karner Blue Butterfly (WI)
- WMS - Drainage (IL)
- WMS - Drainage - (MI)
- WMS - Embankments, Dikes, and Levees
- WMS - Embankments, Dikes, and Levees (OH)
- WMS - Grassed Waterways - (MI)
- AGR - Air Quality; PM10 (TX)
- AGR - Air Quality; PM2_5 (TX)
- AGR - Aronia Berry Suitability (SD)
- AGR - Farmland of Statewide Importance (TX)
- AGR - Index for alfalfa hay, irrigated (NV)
- AGR - Nitrate Leaching Potential, Nonirrigated (MA)
- AGR - Rangeland Grass/Herbaceous Productivity Index (TX)
- AGR - Rutting Hazard > 10,000 Pounds per Wheel (TX)
- AGR - Water Erosion Potential (TX)
- AGR - Wine Grape Site Suitability (WA)
- AGR-Natural Fertility (ND)
- AGR-Subsurface Salinity (ND)
- AWM - Filter Group (OH)
- AWM - Irrigation Disposal of Wastewater
- AWM - Land Application of Dry and Slurry Manure (TX)
- AWM - Land Application of Municipal Biosolids, winter (OR)
- AWM - Overland Flow Process Treatment of Wastewater
- AWM - Rapid Infiltration Disposal of Wastewater
- AWM - Vegetated Treatment Area (PIA)
- AWM - Waste Field Storage Area (VT)
- BLM - Mechanical Treatment, Shredder
- BLM - Medusahead Invasion Susceptibility
- BLM - Soil Compaction Resistance
- Capping Fill Gravity Septic System (DE)
- CLASS RULE - Depth to any bedrock kind (5 classes) (NPS)
- 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 - Grass Hay, NIRR - Palouse, Northern Rocky Mtns. (WA)
- CPI - Small Grains Productivity Index (AK)
- DHS - Catastrophic Event, Large Animal Mortality, Incinerate
- DHS - Emergency Land Disposal of Milk
- DHS - Site for Composting Facility - Subsurface
- DHS - Suitability for Clay Liner Material

- ENG - Cohesive Soil Liner (MN)
- ENG - Construction Materials - Sand Source (MN)
- ENG - Construction Materials; Gravel Source (CT)
- ENG - Construction Materials; Gravel Source (NY)
- ENG - Construction Materials; Reclamation (DE)
- ENG - Construction Materials; Roadfill
- ENG - Construction Materials; Roadfill (AK)
- ENG - Construction Materials; Sand Source (NY)
- ENG - Construction Materials; Sand Source (VT)
- ENG - Construction Materials; Topsoil (AK)
- ENG - Construction Materials; Topsoil (DE)
- ENG - Construction Materials; Topsoil (MI)
- ENG - Construction Materials; Topsoil (OR)
- ENG - Conventional On-Site Septic Systems (TN)
- ENG - Deep Infiltration Systems
- ENG - Disposal Field Gravity (DE)
- ENG - Dwellings With Basements (OH)
- ENG - Ground-based Solar Arrays, Ballast Anchor Systems
- ENG - Large Animal Disposal, Trench (CT)
- ENG - Lawn, Landscape, Golf Fairway (MI)
- ENG - Lawn, Landscape, Golf Fairway (VT)
- ENG - Sanitary Landfill (Area) (OH)
- ENG - Sanitary Landfill (Trench)
- ENG - Sanitary Landfill (Trench) (AK)
- ENG - Septage Application - Surface (MN)
- ENG - Septic Tank Absorption Fields - At-Grade (MN)
- ENG - Septic Tank Absorption Fields - Mound (MN)
- ENG - Septic Tank Leaching Chamber (TX)
- ENG - Septic Tank, Subsurface Drip Irrigation (TX)
- ENG - Shallow Excavations
- ENG - Shallow Infiltration Systems
- ENG - Small Commercial Buildings
- ENG - Soil Potential of Road Salt Applications (CT)
- ENG - Source of Caliche (TX)
- ENG - Stormwater Management / Ponds (NY)
- ENG - Unlined Retention Systems
- Farm and Garden Composting Facility - Surface
- FOR - Biomass Harvest (MA)
- FOR - Black Walnut Suitability Index (KS)
- FOR - Displacement Potential (WA)
- FOR - Drought Vulnerable Soils
- FOR - General Harvest Season (ME)

- FOR - Harvest Equipment Operability
- FOR - Mechanical Site Preparation (Deep) (MD)
- FOR - Mechanical Site Preparation (Surface)
- FOR - Mechanical Site Preparation; Deep (CT)
- FOR - Potential Erosion Hazard (Road/Trail)
- FOR - Potential Fire Damage Hazard
- FOR - Potential Seedling Mortality
- FOR - Potential Seedling Mortality (MI)
- FOR - Potential Windthrow Hazard (ME)
- FOR - Potential Windthrow Hazard (MI)
- FOR - Road Suitability (Natural Surface) (ID)
- FOR - Rutting Hazard by Month
- FOR - Windthrow Hazard (WA)
- FOTG - NLI Interp Calculation - (IN)
- Fragile Soil Index
- GRL - Juniper Encroachment Potential (NM)
- GRL - NV range seeding (Wind C = 20) (NV)
- GRL - Pasture and Hayland SG (OH)
- GRL - Rangeland Prescribed Burning (TX)
- GRL - Rangeland Soil Seed Bank Suitability (NM)
- GRL-FSG-NP-W (MT)
- GRL-SHSI Soil Health Sustainability Index (MT)
- Ground-based Solar Arrays_saturationt(ME)
- Ground-based Solar Arrays_slope(ME)
- Inland Wetlands (CT)
- IRR-restrictive features for irrigation (OH)
- MIL - Excavations for Vehicle Fighting Position (DOD)
- MIL - Trafficability Veh. Type 1 1-pass wet season (DOD)
- MIL - Trafficability Veh. Type 2 dry season (DOD)
- MIL - Trafficability Veh. Type 3 50-passes wet season (DOD)
- MIL - Trafficability Veh. Type 6 1-pass wet season (DOD)
- MIL - Trafficability Veh. Type 6 dry season (DOD)
- Muscadine Wine Grape Site Desirability (Very Long)
- NCCPI - NCCPI Cotton Submodel (II)
- Permafrost Sensitivity (AK)
- Pressure Dose Capping Fill Septic System (DE)
- REC - Camp Areas (CT)
- REC - Off-Road Motorcycle Trails (CT)
- SAS - CMECS Substrate Class
- SAS - CMECS Substrate Subclass/Group
- SAS - Eelgrass Restoration Suitability
- SAS - Land Utilization of Dredged Materials

- SAS - Northern Quahog (Hard Clam) Habitat Suitability
- Septic System At Grade Shallow Field (alternative) (WV)
- Septic System At-Grade Bed (Alternate) (PA)
- Septic System CO-OP RFS III w/Drip Irrigation (PA)
- Septic System Drip Irrigation (alternative) (WV)
- Septic System Free Access Sand Filterw/Spray Irrigation (PA)
- Septic System Peat Based Option1 w/At-Grade Bed (Alt.) (PA)
- Septic System Spray Irrigation (PA)
- Septic System Steep Slope Sand Mound (Alternate) (PA)
- Shallow Infiltration Systems
- SOH - Organic Matter Depletion Potential, Irrigated (CA)
- SOH - Soil Surface Sealing
- TROP - Plantains Productivity
- URB/REC - Camp Areas (GA)
- URB/REC - Camp Areas (MI)
- URB/REC - Golf Fairways (OH)
- URB/REC - Off-Road Motorcycle Trails
- URB/REC - Paths and Trails (MI)
- URB/REC - Playgrounds (OH)
- Vinifera Wine Grape Site Desirability (Long to Medium)
- WLF - Chufa for Turkey Forage (LA)
- WLF - Food Plots for Upland Wildlife < 2 Acres (TX)
- WLF - Freshwater Wetland Plants (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 Mixed Deciduous & Coniferous Trees (TX)
- WMS - Constructing Grassed Waterways (TX)
- WMS - Constructing Terraces and Diversions (OH)
- WMS - Embankments, Dikes, and Levees (VT)
- WMS - Irrigation, Sprinkler (close spaced outlet drops)
- WMS - Irrigation, Sprinkler (general)
- WMS - Pond Reservoir Area (GA)
- WMS-Subsurface Water Management, Installation (ND)
- WMS-Subsurface Water Management, Outflow Quality (ND)
- AGR - Barley Yield (MT)
- AGR - Conventional Tillage (TX)
- AGR - Grape non-irrigated (MO)
- AGR - Industrial Hemp for Fiber and Seed Production
- AGR - Nitrate Leaching Potential, Irrigated (WA)
- AGR - Pasture hayland (MO)

- AGR - Pesticide Loss Potential-Soil Surface Runoff
- AGR - Prime Farmland (TX)
- AGR - Spring Wheat Yield (MT)
- AGR-Agronomic Concerns (ND)
- AGR-Pesticide and Nutrient Leaching Potential, NIRR (ND)
- AGR-Surface Salinity (ND)
- AGR-Water Erosion Potential (ND)
- Alaska Exempt Wetland Potential (AK)
- American Wine Grape Varieties Site Desirability (Short)
- AWM - Irrigation Disposal of Wastewater (MD)
- AWM - Manure and Food Processing Waste (DE)
- AWM - Manure Stacking - Site Evaluation (TX)
- AWM - Phosphorus Management (TX)
- AWM - Slow Rate Process Treatment of Wastewater
- BLM - Pygmy Rabbit Habitat Potential
- BLM - Rangeland Tillage
- BLM - Site Degradation Susceptibility
- CA Prime Farmland (CA)
- CLASS RULE - Depth to root limiting layer (5 classes) (NPS)
- Commodity Crop Productivity Index (Corn) (TN)
- CPI - Alfalfa Hay, NIRR - Palouse, Northern Rocky Mtns. (ID)
- CPI - Barley, NIRR - Eastern Idaho Plateaus (ID)
- CPI - Grass Hay, IRR - Eastern Idaho Plateaus (ID)
- CPI - Grass Hay, NIRR - Palouse, Northern Rocky Mtns. (ID)
- CPI - Potatoes, IRR - Snake River Plains (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, 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 - Site for Composting Facility - Surface
- Elevated Sand Mound Septic System (DE)
- ENG - Animal Disposal by Composting (Catastrophic) (WV)
- ENG - Application of Municipal Sludge (TX)
- ENG - Closed-Loop Horizontal Geothermal Heat Pump (CT)
- ENG - Construction Materials; Gravel Source (IN)
- ENG - Construction Materials; Gravel Source (NE)
- ENG - Construction Materials; Reclamation (MD)
- ENG - Construction Materials; Reclamation (MI)

- ENG - Construction Materials; Roadfill (GA)
- ENG - Construction Materials; Sand Source (CT)
- ENG - Construction Materials; Sand Source (GA)
- ENG - Construction Materials; Topsoil (ID)
- ENG - Construction Materials; Topsoil (OH)
- ENG - Daily Cover for Landfill (OH)
- ENG - Disposal Field (NJ)
- ENG - Disposal Field Type Inst (NJ)
- ENG - Dwellings W/O Basements
- ENG - Dwellings With Basements
- ENG - Dwellings without Basements (AK)
- ENG - Lawn and Landscape (OH)
- ENG - Lawn, Landscape, Golf Fairway
- ENG - Local Roads and Streets (AK)
- ENG - Local Roads and Streets (GA)
- ENG - On-Site Waste Water Lagoons (MO)
- ENG - Pier Beam Building Foundations (TX)
- ENG - Sanitary Landfill (Area)
- ENG - Sanitary Landfill (Area) (AK)
- ENG - Septage Application - Incorporation or Injection (MN)
- ENG - Septic System; Disinfection, Surface Application (TX)
- ENG - Septic Tank Absorption Fields (FL)
- ENG - Septic Tank Absorption Fields (OH)
- ENG - Septic Tank Absorption Fields - Trench (MN)
- ENG - Sewage Lagoons (AK)
- ENG - Shallow Excavations (OH)
- ENG - Soil Suitability for SLAMM Marsh Migration (CT)
- ENG - Stormwater Management / Infiltration (NY)
- ENG - Stormwater Management / Wetlands (NY)
- FOR - Black Walnut Suitability (WI)
- FOR - Black Walnut Suitability (WV)
- FOR - Construction Limitations for Haul Roads/Log Landings
- FOR - Displacement Hazard
- FOR - Harvest Equipment Operability (DE)
- FOR - Harvest Equipment Operability (ME)
- FOR - Harvest Equipment Operability (MI)
- FOR - Log Landing Suitability (ID)
- FOR - Log Landing Suitability (MI)
- FOR - Log Landing Suitability (OR)
- FOR - Mechanical Planting Suitability (OH)
- FOR - Mechanical Site Preparation (Surface) (MD)
- FOR - Mechanical Site Preparation (Surface) (OH)

- FOR - Mechanical Site Preparation; Surface (CT)
- FOR - Potential Erosion Hazard (Off-Road/Off-Trail) (MI)
- FOR - Potential Erosion Hazard (Off-Road/Off-Trail) (OH)
- FOR - Potential Seedling Mortality (FL)
- FOR - Potential Seedling Mortality (OH)
- FOR - Road Suitability (Natural Surface) (VT)
- FOR - Soil Rutting Hazard
- FOTG - Indiana Soy Bean Yield Calculation (IN)
- FOTG - Indiana Wheat Yield Calculation (IN)
- FOTG - NLI report Calculation - (IN)
- GRL - Fencing, Post Depth =<24 inches
- GRL - Fencing, Post Depth Less Than 24 inches (TX)
- GRL - Fencing, Post Depth Less Than 36 inches (TX)
- GRL - NV range seeding (Wind C = 10) (NV)
- GRL - NV range seeding (Wind C = 30) (NV)
- GRL - Rangeland Chaining (TX)
- GRL - Rangeland Disking (TX)
- GRL - Rangeland Dozing/Grubbing (TX)
- GRL - Utah Juniper Encroachment Potential
- GRL - Western Juniper Encroachment Potential (OR)
- Ground-based Solar Arrays_bedrock_slope_anchor(ME)
- Ground-based Solar Arrays_saturation_flooding_Frost(ME)
- Hybrid Wine Grape Varieties Site Desirability (Medium)
- Lined Retention Systems
- MIL - Trafficability Veh. Type 1 dry season (DOD)
- MIL - Trafficability Veh. Type 3 1-pass wet season (DOD)
- MIL - Trafficability Veh. Type 3 dry season (DOD)
- MIL - Trafficability Veh. Type 4 dry season (DOD)
- MIL - Trafficability Veh. Type 5 1-pass wet season (DOD)
- NCCPI - NCCPI Corn Submodel (I)
- NCCPI - NCCPI Small Grains Submodel (II)
- NCCPI - NCCPI Soybeans Submodel (I)
- Peony Flowers Site Suitability (AK)
- Pressure Dose Full Depth Septic System (DE)
- REC - Camp Areas; Primitive (AK)
- REC - Paths and Trails (CT)
- Salinity Risk Index (ND)
- SAS - Eastern Oyster Habitat Restoration Suitability
- SAS - Mooring Anchor - Mushroom
- Septic System CO-OP RFS III w/At-Grade Bed (PA)
- Septic System Free Access Sand Filter w/At-Grade Bed (PA)
- Septic System Modified Subsurface Sand Filter (Alt.) (PA)

- Septic System Shallow In Ground Trench (conventional) (WV)
- Septic System Subsurface Sand Filter Bed (conventional) (PA)
- Septic System Subsurface Sand Filter Trench (standard) (PA)
- SOH - Limitations for Aerobic Soil Organisms
- URB - Concrete Driveways and Sidewalks (TX)
- URB - Dwellings on Concrete Slab (TX)
- URB - Lawns and Ornamental Plantings (TX)
- URB/REC - Paths and Trails
- URB/REC - Paths and Trails (GA)
- URB/REC - Playgrounds (MI)
- Vinifera Wine Grape Site Desirability (Long)
- WLF - Crawfish Aquaculture (TX)
- WLF - Desertic Herbaceous Plants (TX)
- WLF - Gopher Tortoise Burrowing Suitability
- WLF - Grain & Seed Crops for Food and Cover (TX)
- WMS - Constructing Grassed Waterways (OH)
- WMS - Irrigation, Surface (graded)
- WMS - Subsurface Drains - Installation (VT)
- WMS - Subsurface Water Management, System Performance
- WMS - Surface Drains (TX)
- WMS - Surface Irrigation Intake Family (TX)
- Septic System Low Pressure Pipe (alternative) (WV)
- Septic System Mound (alternative) (WV)
- Septic System Peat Based Option2 w/Spray Irrigation (PA)
- Septic System Steep Slope Mound (alternative) (WV)
- SOH - Concentration of Salts- Soil Surface
- SOH - Soil Susceptibility to Compaction
- Soil Habitat for Saprophyte Stage of Coccidioides
- Unlined Retention Systems
- URB - Commercial Metal Bldg; w/Reinforced Concrete Slab (TX)
- URB/REC - Picnic Areas (GA)
- URB/REC - Picnic Areas (MI)
- URB/REC - Picnic Areas (OH)
- Vinifera Wine Grape Site Desirability (Short)
- WLF - Burrowing Mammals & Reptiles (TX)
- WLF - Desert Tortoise (CA)
- WLF - Domestic Grasses & Legumes for Food and Cover (TX)
- WLF - Irrigated Grain & Seed Crops for Food & Cover (TX)
- WMS - Excavated Ponds (Aquifer-fed)
- WMS - Excavated Ponds (Aquifer-fed) (VT)
- WMS - Irrigation, General
- WMS - Irrigation, Micro (above ground)

- WMS - Irrigation, Micro (above ground) (VT)
- WMS - Irrigation, Micro (subsurface drip)
- WMS - Irrigation, Sprinkler (general) (VT)
- WMS - Pond Reservoir Area
- WMS - Pond Reservoir Area (OH)
- WMS - Subsurface Water Management, System Installation
- WMS - Constructing Terraces & Diversions (TX)
- WMS - Drainage (OH)
- WMS - Excavated Ponds (Aquifer-fed) (OH)
- WMS - Grape Production with Drip Irrigation (TX)
- WMS - Irrigation, Micro (subsurface drip) (VT)
- WMS - Irrigation, Surface (level)
- WMS - Pond Reservoir Area (MI)
- WMS - Pond Reservoir Area (VT)
- WMS - Sprinkler Irrigation (MT)
- WMS - Sprinkler Irrigation RDC (IL)
- WMS - Subsurface Drains - Performance (VT)
- WMS - Subsurface Water Management, Outflow Quality
- WMS - Surface Water Management, System
- WMS-Subsurface Water Management, Performance (ND)

Value

a data.frame

Author(s)

Jason Nemecek, Chad Ferguson, Andrew Brown

Examples

```
# get two forestry interpretations for CA630
get_SDA_interpretation(c("FOR - Potential Seedling Mortality",
                         "FOR - Road Suitability (Natural Surface)"),
                       method = "Dominant Condition",
                       areasymbols = "CA630")
```

get_SDA_metrics *Get Soil Data Access, Lab Data Mart and Web Soil Survey Usage Metrics*

Description

Obtain pre-calculated tabular reports of usage, activities, areas of interest (AOI), exports, ecological sites, ratings and reports for specific areas, times and intervals.

Usage

```
get_SDA_metrics(query_name, query_frequency, query_year, state = NULL)
```

Arguments

query_name	One or more of: 'LDM_Usage', 'SDA_Usage', 'wss_ActivityCounts', 'wss_AOIDefinition', 'wss_AOISizeRange', 'wss_ExportCounts', 'wss_PrintableOutput', 'wss_top100AOIs', 'wss_top100Ecologicalsites', 'wss_top100ratings', 'wss_top100reports'
query_frequency	One or more of: 'M', 'CY', 'FY'
query_year	Integer. One or more years e.g. 2020:2021
state	Optional: State abbreviation; Default: NULL uses "xnational" for all states.

Value

A data.frame containing query results

Author(s)

Jason Nemecek

Examples

```
## Not run:
get_SDA_metrics('SDA_Usage', 'CY', 2019:2021)

## End(Not run)
```

`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,
  WHERE = NULL,
  query_string = FALSE,
  dsn = NULL
)
```

Arguments

`areasymbols`	vector of soil survey area symbols
`mukeys`	vector of map unit keys
`WHERE`	character containing SQL WHERE clause specified in terms of fields in `legend`, `mapunit`, or `muaggatt` tables, used in lieu of `mukeys` or `areasymbols`
`query_string`	Default: `FALSE`; if `TRUE` return a character string containing query that would be sent to SDA via `SDA_query`
`dsn`	Path to local SQLite database or a DBIConnection object. If `NULL` (default) use Soil Data Access API via `SDA_query()`.

Value

a data.frame

Author(s)

Jason Nemecek, Chad Ferguson, Andrew Brown

get_SDA_pmgroupname *Get map unit parent material group information from Soil Data Access*

Description

Get map unit parent material group information from Soil Data Access

Usage

```
get_SDA_pmgroupname(
  areasymbols = NULL,
  mukeys = NULL,
  WHERE = NULL,
  method = "DOMINANT COMPONENT",
  simplify = TRUE,
  include_minors = TRUE,
  miscellaneous_areas = FALSE,
  query_string = FALSE,
  dsn = NULL
)
```

Arguments

areasymbols	*character*. Vector of soil survey area symbols
mukeys	*integer*. Vector of map unit keys
WHERE	*character*. SQL WHERE clause specified in terms of fields in legend, mapunit, component, or copmgrp tables, used in lieu of mukeys or areasymbols
method	*character*. One of: "Dominant Component", "Dominant Condition", "None"
simplify	*logical*. Group into generalized parent material groups? Default TRUE
include_minors	logical. Include minor components? Default: TRUE.
miscellaneous_areas	*logical*. Include miscellaneous areas (non-soil components) in results? Default: FALSE.
query_string	Default: FALSE; if TRUE return a character string containing query that would be sent to SDA via SDA_query
dsn	Path to local SQLite database or a DBIConnection object. If NULL (default) use Soil Data Access API via SDA_query().

Details

Default method is "Dominant Component" to get the dominant component (highest percentage). Use "Dominant Condition" or dominant parent material condition (similar conditions aggregated across components). Use "None" for no aggregation (one record per component).

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,
  WHERE = NULL,
  top_depth = 0,
  bottom_depth = 200,
  FUN = NULL,
  include_minors = FALSE,
  miscellaneous_areas = FALSE,
  query_string = FALSE,
  dsn = NULL
)
```

Arguments

`property`	character vector of labels from property dictionary tables (see details) OR physical column names from `component` or `chorizon` table.
`method`	one of: "Dominant Component (Category)", "Dominant Component (Numeric)", "Weighted Average", "MIN", "MAX", "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.
`areasymbols`	vector of soil survey area symbols
`mukeys`	vector of map unit keys
`WHERE`	character containing SQL WHERE clause specified in terms of fields in `legend` or `mapunit` tables, used in lieu of `mukeys` or `areasymbols`. With aggregation method `"NONE"` the WHERE clause may additionally contain logic for columns from the `component` and `chorizon` table.

top_depth	Default: 0 (centimeters); a numeric value for upper boundary (top depth) used only for method="Weighted Average", "Dominant Component (Numeric)", and "MIN/MAX"
bottom_depth	Default: 200 (centimeters); a numeric value for lower boundary (bottom depth) used only for method="Weighted Average", "Dominant Component (Numeric)", and "MIN/MAX"
FUN	Optional: character representing SQL aggregation function either "MIN" or "MAX" used only for method="min/max"; this argument is calculated internally if you specify method="MIN" or method="MAX"
include_minors	Include minor components in "Weighted Average" or "MIN/MAX" results? Default: TRUE
miscellaneous_areas	Include miscellaneous areas (non-soil components) in results? Default: FALSE. Now works with all method types)
query_string	Default: FALSE; if TRUE return a character string containing query that would be sent to SDA via SDA_query
dsn	Path to local SQLite database or a DBIConnection object. If NULL (default) use Soil Data Access API via SDA_query().

Details

The property argument refers to one of the property names or columns specified in the tables below. Note that property can be specified as either a character vector of labeled properties, such as "Bulk Density 0.33 bar H2O - Rep Value", OR physical column names such as "dbthirdbar_r". To get "low" and "high" values for a particular property, replace the _r with _l or _h in the physical column name; for example property = c("dbthirdbar_l","dbthirdbar_r","dbthirdbar_h"). You can view exhaustive lists of component and component horizon level properties in the Soil Data Access "Tables and Columns Report".

Selected Component-level Properties:

Property (Component)	Column
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	hydgrp
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:

Property (Horizon)	Column
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	dbtenthbar_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
Electrial 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 inches - Rep Value	frag3to10_r
Rock Fragments > 10 inches - 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 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

Examples

```
# get 1/3 bar bulk density [0,25] centimeter depth weighted average from dominant component
get_SDA_property(property = c("dbthirdbar_l","dbthirdbar_r","dbthirdbar_h"),
                 method = "Dominant Component (Numeric)",
                 areasymbols = "CA630",
                 top_depth = 0,
                 bottom_depth = 25)
```

---

get_SDV_legend_elements

Get Soil Data Viewer Attribute Information

Description

Get Soil Data Viewer Attribute Information

Usage

```
get_SDV_legend_elements(
  WHERE,
  alpha = 255,
  notratedcolor = rgb(1, 1, 1, 0),
  simplify = TRUE
)
```

Arguments

WHERE	WHERE clause for query of Soil Data Access sdvattribute table
alpha	transparency value applied in calculation of hexadecimal color. Default: 255 (opaque).
notratedcolor	Used to add 'Not rated' color entries where applicable. Default: "#FFFFFF00" (transparent white).
simplify	Return a data.frame when WHERE is length 1? Return a list with 1 element per legend when WHERE is length > 1? Default: TRUE

Value

A list with a data.frame element for each element of WHERE containing "attributekey", "attributename", "attributetype", "attributetablename", "attributecolumnname", "attributedescription", "nasisrulename", "label", "order", "value", "lower_value", "upper_value","red", "green", "blue" and "hex" columns.

---

get_site_data_from_NASIS_db

Get Site Data from a local NASIS Database

Description

Get site-level data from a local NASIS database.

get_site_association_from_NASIS(): Get Associated User Site IDs for each Site.

Usage

```
get_site_data_from_NASIS_db(
  SS = TRUE,
  include_pedon = TRUE,
  nullFragsAreZero = TRUE,
  stringsAsFactors = NULL,
  dsn = NULL
)

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

Arguments

SS	fetch data from Selected Set in NASIS or from the entire local database (default: TRUE)
include_pedon	Include pedon and transect data joined to site? (default: TRUE)
nullFragsAreZero	should surface fragment cover percentages of NULL be interpreted as 0? (default: TRUE)
stringsAsFactors	deprecated
dsn	Optional: path to local SQLite database containing NASIS table structure; default: NULL

Details

It is possible to have multiple pedon records per site observation and multiple site observations per site, which will result in multiple records per site. See argument include_pedon=FALSE to omit joins to pedon and transect tables.

The following aggregations of child table information are performed by this function:

- Site Area Overlap for State, County and MLRA are returned for related area records, as specified in the site table, as the following column names: site_state, site_county, and site_mlra.
- Site Observation Surface Fragment data are simplified (converted to wide format) using simplifyFragmentData().
- The best Ecological Site History record is selected using get_ecosite_history_from_NASIS_db(best = TRUE).
- Site Other Vegetation Class information is aggregated by class name, using " & " as the separator when multiple classes are assigned.
- When multiple Site Bedrock entries are present, only the shallowest is returned by this function. In lieu of bedrock depth the first record in the table is returned.

Value

A data.frame

Author(s)

Jay M. Skovlin, Dylan E. Beaudette, Andrew G. Brown, Greg Schmidt

See Also

get_hz_data_from_NASIS_db(), fetchNASIS(), fetchVegdata()

get_site_data_from_pedon_db

Get 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.

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 Series Classification (SC) database

Description

These functions return records from the Series Classification (SC) database, either from the local NASIS database (all series) or via web report (named series only).
get_competing_soilseries_from_NASIS(): Get Soil Series from NASIS Matching Taxonomic Class Name

Usage

```
get_soilseries_from_NASIS(
  stringsAsFactors = NULL,
  dsn = NULL,
  delimiter = " over ",
  SS = FALSE
)

get_soilseries_from_NASISWebReport(soils, stringsAsFactors = NULL)

get_competing_soilseries_from_NASIS(
  x,
  what = "taxclname",
  dsn = NULL,
  SS = FALSE
)
```

Arguments

stringsAsFactors	deprecated
dsn	Optional: path or *DBIConnection* to local database containing NASIS table structure; default: NULL
delimiter	*character*. Used to collapse taxminalogy records where multiple values are used to describe strongly contrasting control sections. Default " over " creates combination mineralogy classes as they would be used in the family name.
SS	*logical*. Fetch data from the currently loaded selected set in NASIS or from the entire local database (default: FALSE; this is to allow for queries against the full Series Classification database as default)
soils	A vector of soil series names
x	Taxonomic Class Name (or other field specified by what) to match, use % for wildcard
what	Column name to match x against, default: 'taxclname'

Value

A data.frame

Author(s)

Stephen Roecker

get_SRI *Get Soil Inventory Resource (SRI) for USFS Region 6*

Description

This function calls ECOSHARE (zip files) to get Soil Inventory Resource (SRI) data for USFS Region 6. These datasets contain both spatial and non-spatial data in the form of a File Geodatabase (GDB).

Usage

```
get_SRI(gdb, layers = "MapUnits", quiet = FALSE, simplify = TRUE)
```

Arguments

gdb	A character of the GDB, e.g. 'Deschutes'.
layers	A character of the layer(s) within the GDB, e.g. 'MapUnits' (default).
quiet	A logical; suppress info on name, driver, size and spatial reference, or signaling no or multiple layers.
simplify	A logical; whether to return a simplified list (data.frame or sf) if length(layers) == 1.

Details

Due to the fact that many Region 6 Forests do not have NRCS SSURGO surveys (at a scale of 1:24,000, these are the highest-resolution soils data generally available), Region 6 initiated a project in 2012 to bring these legacy SRI soils data into digital databases to facilitate their use in regional planning activities. The datasets available on this page are the results of that effort.

The SRI were originally compiled in 20 volumes, with the original year of publication ranging from 1969 to 1979. The Gifford-Pinchot SRI was redone following the eruption of Mt Saint Helens, and that version was published in 1992. The Olympic NF also produced two versions, the original version being published in 1969, with an update in 1982. The Colville National Forest was the only Region 6 forest that did not compile a SRI.

The data are organized into one single regional GDB, together with twenty individual forest-level GDBs. The regional database contains polygons from all twenty SRIs together with a common set of attributes for the two or three soil layers delineated in the individual mapping unit descriptions, such as texture, depth, color, rock content, etc. In general, the regional database contains physical soil attributes that could be compiled more or less completely and consistently across all forests. The individual forest-level databases contain the polygons for each individual SRI, together with various tables of management interpretations and laboratory data, together with a variety of miscellaneous tables. The information contained in these forest-level databases varies widely from forest to forest, which is why they were not merged into a regional view. Full metadata are included with each database, and scans of the original SRI volumes are provided for reference as well. A Forest Service General Technical Report that fully describes the available data is currently in preparation.

The GDB's currently available:

- **Region6**
- **Deschutes**
- **Fremont**
- **GiffordPinchot**
- **Malheur**
- **MtBaker**
- **MtHood**
- **Ochoco**
- **Okanogan**
- **Olympic**
- **RogueRiver**
- **Siskiyou**
- **Siuslaw**
- **Umatilla**
- **Umpqua**
- **WallowaWhitman**
- **Wenatchee**
- **Willamette**
- **Winema**

Value

An `sf` or `data.frame` object.

Note

Please use `get_SRI_layers` to get the layer id information needed for the layer argument. This will help with joining `sf` and `data.frame` objects.

Author(s)

Josh Erickson

See Also

`get_SRI_layers()`

Examples

```
## Not run:

# get Deschutes SRI
sri_deschutes <- get_SRI('Deschutes')

# get multiple layers in a list

sri_deschutes_multiple <- get_SRI(gdb = 'Deschutes',
layers = c('MapUnits', 'ErosionAndHydro', 'SampleSites_MaterialsTesting'))


## End(Not run)
```

`get_SRI_layers` *Get SRI Layers*

Description

Get SRI Layers

Usage

```
get_SRI_layers(gdb)
```

Arguments

gdb	A `character` of the GDB, e.g. `'Deschutes'`.

Value

A list of metadata about the GDB

Note

Refer to `get_SRI` for information on File Geodatabase (GDB) availability.

Author(s)

Josh Erickson

Examples

```
## Not run:
sri_layers <- get_SRI_layers('Willamette')

## End(Not run)
```

get_text_notes_from_NASIS_db *Get text note data from a local NASIS Database*

Description

Get text note data from a local NASIS Database

Usage

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

get_mutext_from_NASIS_db(SS = TRUE, fixLineEndings = TRUE, dsn = NULL)

get_cotext_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 *Get vegetation data from a local NASIS Database*

Description

Get vegetation data from a local NASIS Database. Result includes two data.frames corresponding to the "Plot Plant Inventory" and "Vegetation Transect" child tables of "Vegetation Plot".

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

Value

A list of data.frame

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 *Get Vegetation Data from an AK Site Database*

Description

Get 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.

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 *Get 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).

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

Get 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

Get 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).

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

Get 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).

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` *Get 800m gridded soil properties from SoilWeb 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 `Spatial*`, `RasterLayer`, `sf`, `sfc` or `bbox` object, OR a `list`, see details
`var`	ISSR-800 grid name (case insensitive), see details
`res`	grid resolution, units of meters. The native resolution of ISSR-800 grids (this WCS) is 800m.
`quiet`	logical, passed to `curl::curl_download` to enable / suppress URL and progress bar for download.

Details

`aoi` should be specified as a `SpatRaster`, `Spatial*`, `RasterLayer`, `SpatRaster/SpatVector`, `sf`, `sfc`, or `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. 'OGC:CRS84' (EPSG:4326, WGS84 Longitude/Latitude)

The WCS query is parameterized using a rectangular extent derived from the above AOI specification, after conversion to the native CRS (EPSG:5070) of the ISSR-800 grids.

Variables available from this WCS can be queried using `WCS_details(wcs = 'ISSR800')`.

Value

A SpatRaster (or RasterLayer) object containing indexed map unit keys and associated raster attribute table or a try-error if request fails. By default, spatial classes from the `terra` package are returned. If the input object class is from the `raster` or `sp` packages a RasterLayer is returned.

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

Examples

```
## Not run:
library(terra)

# see WCS_details() for variable options
WCS_details(wcs = 'ISSR800')

# get wind erodibility group
res <- ISSR800.wcs(list(aoi = c(-116, 35, -115.5, 35.5), crs = "EPSG:4326"),
                   var = 'weg', res = 800)
plot(res)

## End(Not run)
```

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^-6, phi_max = 10^8, 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 1/cm

npar index of pore size distribution, function expects log10-transformed values (dimensionless)

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

```
library(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
aqp::plotSPC(gopheridge[1:30, ], name='hzname', color='soil_color')
aqp::plotSPC(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
aqp::plotSPC(gopheridge[1:30, ], name='hzname', color='clay')
aqp::plotSPC(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
aqp::plotSPC(gopheridge, name='', print.id=FALSE, plot.order=order(gopheridge$bedrckdepth))

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

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

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

## loafercreek
data("loafercreek")

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

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

# add diagnostic horizons
aqp::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 a NASIS data source. This function *always* returns FALSE when the odbc package is not available (regardless of whether you have an ODBC data source properly set up).

Usage

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

Arguments

dsn	Optional: path to local SQLite database, or a DBIConnection, containing NASIS table structure; default: NULL

Details

If dsn is specified as a character vector it is assumed to refer to a SQLite data source. The result will be TRUE or FALSE depending on the result of RSQLite::dbCanConnect().

If dsn is specified as a DBIConnection the function returns the value of DBI::dbExistsTable("MetadataDomainMaster")

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 Ecological Dynamics Interpretive Tool (EDIT) web services URL*

Description

Construct a URL for Ecological Dynamics Interpretive Tool (EDIT) web services (`https://edit.jornada.nmsu.edu/serv` 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`, `soil-features.json`
`querystring`	Optional: Additional request parameters specified as a query string `?param1=value&param2=value`.

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)
```

metadata *NASIS 7 Metadata*

Description

NASIS 7 Metadata from MetadataDomainDetail, MetadataDomainMaster, and MetadataTableColumn tables

Format

A `data.frame` with the following columns:

- `DomainID` - Integer. ID that uniquely identifies a domain in a data model, not just within a database.
- `DomainName` - Character. Domain Name.
- `DomainRanked` - Integer. Is domain ranked? `0` = No; `1` = Yes
- `DisplayLabel` - Character. Domain Display Label.
- `ChoiceSequence` - Integer. Order or sequence of Choices.
- `ChoiceValue` - Integer. Value of choice level.
- `ChoiceName` - Character. Name of choice level.
- `ChoiceLabel` - Character. Label of choice level.
- `ChoiceObsolete` - Integer. Is choice level obsolete? `0` = No; `1` = Yes
- `ColumnPhysicalName` - Character. Physical column name.
- `ColumnLogicalName` - Character. Logical column name.

mukey.wcs *Get Map Unit Key (*`mukey`*) grid from SoilWeb Web Coverage Service (WCS)*

Description

Download chunks of the gNATSGO, gSSURGO, RSS, and STATSGO2 map unit key grid via bounding-box from the SoilWeb WCS.

Usage

```
mukey.wcs(
  aoi,
  db = c("gNATSGO", "gSSURGO", "RSS", "STATSGO", "PR_SSURGO", "HI_SSURGO"),
  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', 'gSSURGO', 'STATSGO', 'HI_SSURGO', or 'PR_SSURGO' (case insensitive)
res	grid resolution, units of meters. The native resolution of gNATSGO and gSSURGO (this WCS) is 30m; STATSGO (this WCS) is 300m; and Raster Soil Surveys (RSS) are at 10m resolution. If res is not specified the native resolution of the source is used.
quiet	logical, passed to curl::curl_download to enable / suppress URL and progress bar for download.

Details

aoi should be specified as one of: SpatRaster, 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. 'OGC:CRS84' (EPSG:4326, WGS84 Longitude/Latitude)

The WCS query is parameterized using a rectangular extent derived from the above AOI specification, after conversion to the native CRS (EPSG:5070) of the WCS grids.

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

Value

A SpatRaster (or RasterLayer) object containing indexed map unit keys and associated raster attribute table or a try-error if request fails. By default, spatial classes from the terra package are returned. If the input object class is from the raster or sp packages a RasterLayer is returned.

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

Examples

```
## Not run:
library(terra)

res <- mukey.wcs(list(aoi = c(-116.7400, 35.2904, -116.7072, 35.3026), crs = "EPSG:4326"),
                 db = 'gNATSGO', res = 30)
```

```
m <- unique(values(res))

prp <- setNames(
  get_SDA_property(
    c("ph1to1h2o_r", "claytotal_r"),
    "weighted average",
    mukeys = m,
    top_depth = 0,
    bottom_depth = 25,
    include_minors = TRUE,
    miscellaneous_areas = FALSE
  )[, c("mukey", "ph1to1h2o_r", "claytotal_r")],
  c("ID",    "pH1to1_0to25", "clay_0to25")
)

levels(res) <- prp
res2 <- catalyze(res)
res2

plot(res2[['pH1to1_0to25']])

## End(Not run)
```

NASISChoiceList *Work with NASIS Choice Lists*

Description

Create (ordered) factors and interchange between choice names, values and labels for lists of input vectors.

Usage

```
NASISChoiceList(
  x = NULL,
  colnames = names(x),
  what = "ColumnPhysicalName",
  choice = c("ChoiceName", "ChoiceValue", "ChoiceLabel"),
  obsolete = FALSE,
  factor = TRUE,
  droplevels = FALSE,
  ordered = TRUE,
  simplify = TRUE,
  dsn = NULL
)
```

Arguments

x	A named list of vectors to use as input for NASIS Choice List lookup

colnames	vector of values of the column specified by what. E.g. colnames="texcl" for what="ColumnPhysicalName". Default: names(x) (if x is named)
what	passed to get_NASIS_column_metadata(); Column to match x against. Default "ColumnPhysicalName"; alternate options include "DomainID", "DomainName", "DomainRanked", "DisplayLabel", "ChoiceSequence", "ChoiceValue", "ChoiceName", "ChoiceLabel", "ChoiceObsolete", "ChoiceDescription", "ColumnLogicalName"
choice	one of: "ChoiceName", "ChoiceValue", or "ChoiceLabel"
obsolete	Include "obsolete" choices? Default: FALSE
factor	Convert result to factor? Default: TRUE
droplevels	Drop unused factor levels? Default: TRUE (used only when factor=TRUE)
ordered	Should the result be an ordered factor? Default: TRUE (use *only* if DomainRanked is true for all choices)
simplify	Should list result with length 1 be reduced to a single vector? Default: TRUE
dsn	Optional: path or *DBIConnection* to local database containing NASIS table structure; default: NULL

Value

A list of "choices" based on the input x that have been converted to a consistent target set of levels (specified by choice) via NASIS 7 metadata.

When factor=TRUE the result is a factor, possibly ordered when ordered=TRUE and the target domain is a "ranked" domain (i.e. ChoiceSequence has logical meaning).

When factor=FALSE the result is a character or numeric vector. Numeric vectors are always returned when choice is "ChoiceValue".

Examples

```
NASISChoiceList(1:3, "texcl")

NASISChoiceList(1:3, "pondfreqcl")

NASISChoiceList("Clay loam", "texcl", choice = "ChoiceValue")

NASISChoiceList("Silty clay loam", "texcl", choice = "ChoiceName")
```

NASISDomainsAsFactor *Get/Set Options for Encoding NASIS Domains as Factors*

Description

Set package option soilDB.NASIS.DomainsAsFactor for returning coded NASIS domains as factors.

Usage

```
NASISDomainsAsFactor(x = NULL)
```

Arguments

x	logical; default FALSE

Value

logical, result of `getOption("soilDB.NASIS.DomainsAsFactor")`

Examples

```
## Not run:
NASISDomansAsFactor(TRUE)

## End(Not run)
```

NASISLocalDatabase *NASIS Local Database*

Description

This is a guide on using databases that follow the NASIS schema. Most of the time users are querying an instance of the Microsoft SQL Server NASIS local transactional database running on their computer. It is possible to create file-based "snapshots" of a local instance of the NASIS database using SQLite. See `[createStaticNASIS()]` for details. These file-based snapshots, or other custom connections, can generally be specified to NASIS-related functions via the `dsn` argument.

Working With Coded Values and Decoding

Some values (choice lists) in NASIS are conventionally stored using numeric codes. The codes are defined by "domain" and allow for both "names" and "labels" as well as other descriptive information to be provided for each choice list element. See `get_NASIS_column_metadata()` for details.

Many soilDB functions call the function `uncode()` internally to handle conversion to human-readable values using official NASIS domains. If writing queries directly against the database source, such as a connection created with `NASIS()` or query run with `dbQueryNASIS()`, you call `uncode()` on the *data.frame* result of your query. Conversion of internal values to choice list names is based on domains associated with result column names.

When using a custom SQLite database, sometimes values in the database are delivered pre-decoded to make the database more directly usable. An example of this would be the Kellogg Soil Survey Laboratory morphologic database, the NASIS data corresponding to the laboratory analyses available through the Lab Data Mart (LDM).

To avoid issues with offsets between internal storage value and external readable value (for data such as farmland classification or Munsell color value and chroma), you should not call `uncode()` multiple times. Also, you can disable the "decoding" behavior made internally in soilDB functions by setting `options(soilDB.NASIS.skip_uncode = TRUE)`.

NASIS_table_column_keys *NASIS 7 Tables, Columns and Foreign Keys*

Description

This dataset contains NASIS 7 Tables, Columns and Foreign Keys

OSDquery *Search full text of Official Series Description on SoilWeb*

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

`ric`	search range in characteristics section
`use_and_veg`	search use and vegetation section
`competing_series`	search competing series section
`geog_location`	search geographic setting section
`geog_assoc_soils`	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

USDA-NRCS OSD search tools: `https://soilseries.sc.egov.usda.gov/`

See Also

`fetchOSD`, `siblings`, `fetchOSD`

Examples

```
# 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))
aqp::plotSPC(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 `index`, otherwise a `list`

Note

Most web reports are for internal use only.

Author(s)

D.E. Beaudette and S.M. Roecker

processSDA_WKT *Post-process Well-Known Text from Soil Data Access*

Description

This is a helper function commonly used with SDA_query to extract WKT (well-known text) representation of geometry to an sf or sp object.

Usage

```
processSDA_WKT(d, g = "geom", crs = 4326, p4s = NULL, as_sf = TRUE)
```

Arguments

d	data.frame returned by SDA_query, containing WKT representation of geometry
g	name of column in d containing WKT geometry
crs	CRS definition (e.g. an EPSG code). Default 4326 for WGS84 Geographic Coordinate System
p4s	Deprecated: PROJ4 CRS definition
as_sf	Return an sf data.frame? If FALSE return a Spatial* object.

Details

The SDA website can be found at https://sdmdataaccess.nrcs.usda.gov. See the SDA Tutorial for detailed examples.

The SDA website can be found at https://sdmdataaccess.nrcs.usda.gov. See the SDA Tutorial for detailed examples.

Value

An sf object or if as_sf is FALSE a Spatial* object.

Note

This function requires the sf package.

Author(s)

D.E. Beaudette, A.G. Brown

ROSETTA *Query USDA-ARS 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"),
  include.sd = FALSE,
  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.
include.sd	logical, include bootstrap standard deviation for estimated parameters
chunkSize	number of records per API call
conf	configuration passed to httr::POST() such as verbose().

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 percent
- optional, bulk density (any moisture basis): mass per volume after accounting for >2mm fragments, units of gm/cm3
- optional, volumetric water content at 33 kPa: roughly "field capacity" for most soils, units of cm^3/cm^3
- optional, volumetric water content at 1500 kPa: roughly "permanent wilting point" for most plants, units of cm^3/cm^3

The Rosetta pedotransfer function predicts five parameters for the van Genuchten model of unsaturated soil hydraulic properties

- theta_r : residual volumetric water content
- theta_s : saturated volumetric water content
- log10(alpha) : retention shape parameter `[log10(1/cm)]`
- log10(npar) : retention shape parameter
- log10(ksat) : saturated hydraulic conductivity `[log10(cm/d)]`

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: doi: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: doi: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: doi: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: https://pypi.org/project/rosetta-soil/.

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.

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: doi: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: doi: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: doi:10.1016/j.jhydrol.2017.01.004.

`SCAN_SNOTEL_metadata` *USDA-NRCS Station Metadata for SCAN, CSCAN, SNOTEL, SNOWLITE Networks*

Description

These metadata are a work in progress.

Format

A `data.frame` with 1186 SCAN, CSCAN, SNOTEL, and SNOWLITE station metadata records

---

`SDA_query` *Query Soil Data Access*

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 and 32Mb JSON serialization limit per query. Queries should contain a WHERE clause 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. All usages of `SDA_query()` should handle the possibility of a `try-error` result in case the web service connection is down or if an invalid query is passed to the endpoint.

Usage

```
SDA_query(q, dsn = NULL)
```

Arguments

q	character. A valid T-SQL query surrounded by double quotes.
dsn	character. Default: NULL uses Soil Data Access remote data source via REST API. Alternately, dsn may be a file path to an SQLite database using the SSURGO schema, or a DBIConnection that has already been created.

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/limsreport.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 for queries that return a single table. A list of data.frame for queries that return multiple tables. NULL if result is empty, and try-error on error.

Note

This function requires the httr, jsonlite, and xml2 packages

Author(s)

D.E. Beaudette, A.G Brown

See Also

SDA_spatialQuery()

Examples

```
## 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, comppct_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, comppct_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
p <- wk::as_wkt(wk::rct(-120.9, 37.7, -120.8, 37.8))
q <- paste0("SELECT mukey, cokey, compname, comppct_r FROM component
    WHERE mukey IN (SELECT DISTINCT mukey FROM
    SDA_Get_Mukey_from_intersection_with_WktWgs84('", p,
     "')) ORDER BY mukey, cokey, comppct_r DESC")

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

SDA_spatialQuery *Query Soil Data Access by spatial intersection with supplied geometry*

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,
  geomAcres = TRUE,
  db = c("SSURGO", "STATSGO", "SAPOLYGON"),
  byFeature = FALSE,
  idcol = "gid",
  query_string = FALSE,
  as_Spatial = getOption("soilDB.return_Spatial", default = FALSE)
)
```

Arguments

geom	an sf or 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, 'mupolygon', 'mupoint', 'muline' overlapping or intersecting map unit polygons, points or lines from selected database, "featpoint'or'featline'' for special feature points and lines, 'areasymbol'': 'data.frame' with intersecting soil survey areas, ''sapolygon'': overlapping or intersecting soil survey area polygons (SSURGO only)
geomIntersection	logical; FALSE (default): overlapping map unit polygons returned, TRUE: intersection of geom + map unit polygons is returned.
geomAcres	logical; TRUE (default): calculate acres of result geometry in column "area_ac" when what returns a geometry column. FALSE does not calculate acres.
db	a character vector identifying the Soil Geographic Databases ('SSURGO' or 'STATSGO') to query. Option *STATSGO* works with what = "mukey" and what = "mupolygon".
byFeature	Iterate over features, returning a combined data.frame where each feature is uniquely identified by value in idcol. Default FALSE.
idcol	Unique IDs used for individual features when byFeature = TRUE; Default "gid"
query_string	Default: FALSE; if TRUE return a character string containing query that would be sent to SDA via SDA_query
as_Spatial	Return sp classes? e.g. Spatial*DataFrame. Default: FALSE.

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 = "mupolygon".

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 an sf object. A try-error in the event the request cannot be made or if there is an error in the query.

Note

Row-order is not preserved across features in `geom` and returned object. Use `byFeature` argument to iterate over features and return results that are 1:1 with the inputs. Polygon area in acres is computed server-side when `what = 'mupolygon'` and `geomIntersection = TRUE`.

Author(s)

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

See Also

SDA_query

Examples

```
## Not run:
library(aqp)
library(sf)

## query at a point

# example point
p <- sf::st_as_sf(data.frame(x = -119.72330, y = 36.92204),
                  coords = c('x', 'y'),
                  crs = 4326)

# 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)

# get commonly used map unit / component / chorizon records
# as a SoilProfileCollection object
# request that results contain `mukey` with `duplicates = TRUE`
x <- fetchSDA(sql, duplicates = TRUE)

# safely set texture class factor levels
# by making a copy of this column
# this will save in lieu of textures in the original
# `texture` column
aqp::horizons(x)$texture.class <- factor(x$texture, levels = aqp::SoilTextureLevels())

# graphical depiction of the result
aqp::plotSPC(
  x,
  color = 'texture.class',
  label = 'compname',
```

```
  name = 'hzname',
  cex.names = 1,
  width = 0.25,
  plot.depth.axis = FALSE,
  hz.depths = TRUE,
  name.style = 'center-center'
)

## query mukey + geometry that intersect with a bounding box

# define a bounding box: xmin, xmax, ymin, ymax
#
#         +-------------------(ymax, xmax)
#         |                         |
#         |                         |
#     (ymin, xmin) -----------------+
b <- c(-119.747629, -119.67935, 36.912019, 36.944987)

# convert bounding box to WKT
bbox.sp <- sf::st_as_sf(wk::rct(
  xmin = b[1],
  xmax = b[2],
  ymin = b[3],
  ymax = b[4],
  crs = sf::st_crs(4326)
))

# results contain associated map unit keys (mukey)
# return SSURGO polygons, after intersection with provided BBOX
ssurgo.geom <- SDA_spatialQuery(bbox.sp,
                                what = 'mupolygon',
                                db = 'SSURGO',
                                geomIntersection = TRUE)

# return STATSGO polygons, after intersection with provided BBOX
statsgo.geom <- SDA_spatialQuery(bbox.sp,
                                 what = 'mupolygon',
                                 db = 'STATSGO',
                                 geomIntersection = TRUE)

# inspect results
par(mar = c(0, 0, 3, 1))
plot(sf::st_geometry(ssurgo.geom), border = 'royalblue')
plot(
  sf::st_geometry(statsgo.geom),
  lwd = 2,
  border = 'firebrick',
  add = TRUE
)
plot(sf::st_geometry(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
aqp::groupedProfilePlot(
  x,
  groups = 'nationalmusym',
  label = 'compname',
  color = 'awc_r',
  col.label = 'Available Water Holding Capacity (cm / cm)',
  alt.label = 'comppct_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
)

## End(Not run)
```

---

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 (`sf` object) or gridded representation of area proportion falling within 800m cells (`SpatRaster` 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:5070`).

Usage

```
seriesExtent(
  s,
  type = c("vector", "raster"),
  timeout = 60,
  as_Spatial = getOption("soilDB.return_Spatial", default = FALSE)
)
```

Arguments

`s`	a soil series name, case-insensitive
`type`	series extent representation, `'vector'`: results in an `sf` object and `'raster'` results in a `SpatRaster` object
`timeout`	time that we are willing to wait for a response, in seconds
`as_Spatial`	Return sp (`SpatialPolygonsDataFrame`) / raster (`RasterLayer`) classes? Default: `FALSE`.

Value

An R spatial object, class depending on `type` and `as_Spatial` arguments

Author(s)

D.E. Beaudette

References

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

Examples

```
## Not run:

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

  # return an sf object
  x <- seriesExtent(s, type = 'vector')
```

```
  # return a terra SpatRasters
  y <- seriesExtent(s, type = 'raster')

  library(terra)
  if (!is.null(x) && !is.null(y)) {
    x <- terra::vect(x)
    # note that CRS are different
    terra::crs(x)
    terra::crs(y)

    # transform vector representation to CRS of raster
    x <- terra::project(x, terra::crs(y))

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

## End(Not run)
```

`siblings` *Get "siblings" and "cousins" for a given soil series*

Description

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

The siblings of any given soil series are defined as those soil components (major and minor) that share a parent map unit with the named series (as a major component). Component names are filtered using a snapshot of the Soil Classification database to ensure that only valid soil series names are included. Cousins are siblings of siblings. Data are sourced from SoilWeb which maintains a copy of the current SSURGO snapshot. Visualizations of soil "siblings"-related concepts can be found in the "Sibling Summary" tab of Soil Data Explorer app: https://casoilresource.lawr.ucdavis.edu/sde/.

Additional resources:

- Soil Series Query Functions
- Soil "Siblings" Tutorial
- SSSA 2019 Presentation - Mapping Soilscapes Using Soil Co-Occurrence Networks

Usage

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

Arguments

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

Value

A list containing:

- sib: data.frame containing siblings, major component flag, and number of co-occurrences
- sib.data: data.frame containing sibling component data (only when component.data = TRUE)
- cousins: data.frame containing cousins, major component flag, and number of co-occurrences (only when cousins = TRUE)
- cousin.data: data.frame containing cousin component data (only when cousins = TRUE, component.data = TRUE)

Author(s)

D.E. Beaudette

References

O'Geen, A., Walkinshaw, M. and Beaudette, D. (2017), SoilWeb: A Multifaceted Interface to Soil Survey Information. Soil Science Society of America Journal, 81: 853-862. doi:10.2136/sssaj2016.11.0386n

See Also

OSDquery, siblings, fetchOSD

Examples

```
# 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
```

simplifyArtifactData *Simplify Coarse Fraction Data*

Description

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

Usage

```
simplifyArtifactData(
  art,
  id.var,
  vol.var = "huartvol",
  nullFragsAreZero = nullFragsAreZero,
  ...
)

simplifyFragmentData(
  rf,
  id.var,
  vol.var = "fragvol",
  prefix = "frag",
  nullFragsAreZero = TRUE,
  msg = "rock fragment volume",
  ...
)
```

Arguments

art	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
vol.var	character vector with the name of the column containing the coarse fragment volume. Default "fragvol" or "huartvol".
nullFragsAreZero	should fragment volumes of NULL be interpreted as 0? (default: TRUE), see details
...	Additional arguments passed to sieving function (e.g. sieves a named numeric containing sieve size thresholds with class name)
rf	a data.frame object, typically returned from NASIS, see details
prefix	a character vector prefix for input
msg	Identifier of data being summarized. Default is "rock fragment volume" but this routine is also used for "surface fragment cover"

Details

This function is mainly intended for processing of NASIS pedon/component 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. Likewise, `simplifyArtifactData` will sieve out human artifacts, and split total volume into "cohesive", "penetrable", "innocuous", and "persistent".

These functions can be applied to data sources other than NASIS by careful use of the `id.var` and `vol.var` arguments.

- `rf` must contain rock or other fragment volumes in the column "fragvol" (or be specified with `vol.var`), fragment size (mm) in columns "fragsize_l", "fragsize_r", "fragsize_h", fragment cementation class in "fraghard" and flat/non-flat in "fragshp".
- `art` must contain artifact volumes in the column "huartvol" (or be specified with `vol.var`), fragment size (mm) in columns "huartsize_l", "huartsize_r", "huartsize_h", artifact cementation class in "huarthard" and flat/non-flat in "huartshp".

Examples:

- KSSL data

Author(s)

D.E. Beaudette, A.G Brown

`simplifyColorData` *Simplify Color Data by ID*

Description

Simplify multiple Munsell color observations associated with each horizon.

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.

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 "colormoiststst".

Examples:

- KSSL data
- soil color mixing tutorial

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 aqp::col2Munsell()

Author(s)

D.E. Beaudette

---

soilColor.wcs *Get 30m or 270m gridded soil soil color data from SoilWeb Web Coverage Service (WCS)*

Description

Moist soil colors, 2022.

Usage

```
soilColor.wcs(aoi, var, res = 270, quiet = FALSE)
```

Arguments

aoi	area of interest (AOI) defined using a Spatial*, RasterLayer, sf, sfc or bbox object, OR a list, see details
var	soil color grid name (case insensitive), see details
res	grid resolution, units of meters, typically '270', or '30', depending on var. See details.
quiet	logical, passed to curl::curl_download to enable / suppress URL and progress bar for download.

Details

aoi should be specified as a SpatRaster, Spatial*, RasterLayer, SpatRaster/SpatVector, sf, sfc, or 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. 'OGC:CRS84' (EPSG:4326, WGS84 Longitude/Latitude)

The WCS query is parameterized using a rectangular extent derived from the above AOI specification, after conversion to the native CRS (EPSG:5070) of the soil color grids.

Variables available from this WCS can be queried using WCS_details(wcs = 'soilColor'). The full resolution version of the soil color grids use a hr suffix, e.g. 'sc025cm_hr'.

Value

A SpatRaster (or RasterLayer) object containing indexed map unit keys and associated raster attribute table or a try-error if request fails. By default, spatial classes from the terra package are returned. If the input object class is from the raster or sp packages a RasterLayer is returned.

Author(s)

D.E. Beaudette and A.G. Brown

Examples

```
## Not run:
library(terra)

# see WCS_details() for variable options
WCS_details(wcs = 'soilColor')

# moist soil color at 25cm, 270m version
res <- soilColor.wcs(list(aoi = c(-116, 35, -115.5, 35.5), crs = "EPSG:4326"),
                     var = 'sc025cm', res = 270)

# note colors and other metadata are stored
# in raster attribute table
plot(res, col = cats(res)[[1]]$col, axes = FALSE, legend = FALSE)

## End(Not run)
```

---

soilDB.env *Get the soilDB environment used for storing error messages and quality control output*

Description

The soilDB package uses an environment to store variables that are created as side effects of various data access and processing routines. `get_soilDB_env()` provides a method to access this environment from the global (user) environment.

Usage

```
soilDB.env

get_soilDB_env()
```

Format

An object of class `environment` of length 0.

Value

a `environment` object

Examples

```
get_soilDB_env()
```

`soilDB_user_dir` *Get soilDB User Directory Path*

Description

Wrapper around `tools::R_user_dir()` for soilDB-specific file storage in a platform-specific, user-level directory. Requires R >= 4.0.

These directories can be used for storing assets used as input to other soilDB functions, or for caching results to allow for offline use or fewer repeated requests to remote sources.

Use argument `remove=TRUE` to delete files stored in the specified directories.

Usage

```
soilDB_user_dir(
  which = c("data", "config", "cache"),
  ...,
  create = TRUE,
  remove = FALSE,
  fsep = .Platform$file.sep,
  mustWork = NA
)
```

Arguments

which	*character*. One of: "data", "config", or "cache". Passed to tools::R_user_dir().
...	*character*. Sub-directories. Additional arguments are passed to file.path()
create	*logical*. When TRUE (default) the directory path is created (recursively).
remove	*logical*. When TRUE the contents of the folder(s) are deleted (recursively). Default: FALSE.
fsep	*character*. File path separator. Passed to file.path() and normalizePath(). Default: .Platform$file.sep
mustWork	*logical*. Throw error if path cannot be normalized; passed to normalizePath(). Default NA generates a warning.

Details

If the directory path does not exist it is created (recursively) when create=TRUE. Output paths are normalized using normalizePath() and the specified path separator (fsep).

Value

character. File paths within specified soilDB user directory. When remove=TRUE result is NULL.

Author(s)

Andrew Gene Brown

Examples

```
soilDB_user_dir("data", c("dataset1", "dataset2"), "source", create = FALSE)
```

SoilWeb_spatial_query *Get SSURGO Data via Spatial Query*

Description

Get SSURGO Data via Spatial Query to SoilWeb

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.

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

Value

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

Note

SDA now supports spatial queries, consider using `SDA_spatialQuery()` instead.

Author(s)

D.E. Beaudette

Examples

```
# 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)
```

---

summarizeSoilTemperature

Get data from 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

```
summarizeSoilTemperature(soiltemp.data)

month2season(x)

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

Arguments

soiltemp.data	A data.frame containing soil temperature data
x	character vector containing month abbreviation e.g. c('Jun', 'Dec', 'Sep')
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: "hour" (if available), "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
tz	Used for custom timezone. Default "" is current locale

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 >= 365 days of non-missing data).

See:

- Henry Mount Soil Climate Database
- fetchHenry Tutorial

Value

a list containing:

sensors	a sf data.frame 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

`taxaExtent` *Get SoilWeb 800m Major Component Soil Taxonomy Grids*

Description

This function downloads a generalized representation of the geographic extent of any single taxon from the top 4 levels of Soil Taxonomy, or taxa matching a given formative element used in Great Group 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(2011) coordinate reference system (EPSG: 5070).

Usage

```
taxaExtent(
  x,
  level = c("order", "suborder", "greatgroup", "subgroup"),
  formativeElement = FALSE,
  timeout = 60,
  as_Spatial = getOption("soilDB.return_Spatial", default = FALSE)
)
```

Arguments

`x`	single taxon label (e.g. `haploxeralfs`) or formative element (e.g. `pale`), case-insensitive
`level`	the taxonomic level within the top 4 tiers of Soil Taxonomy, one of `'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
`as_Spatial`	Return raster (`RasterLayer`) classes? Default: `FALSE`.

Details

See the Geographic Extent of Soil Taxa tutorial for more detailed examples.

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 parentheses)

- 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
- quartzi: (quartzi) 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 typic subgroup, evidence of stratification
- fragiaquic: (fragiaquic) presence of fragic properties, wetter than typic subgroup
- fragic: (fragic) presence of fragic properties
- glacic: (glacic) presence of ice lenses or wedges
- glossaquic: (glossaquic) interfingered horizon boundaries, wetter than typic subgroup
- glossic: (glossic) interfingered 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 typic 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 typic 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 typic 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: (sombric) 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 `SpatRaster` object (or `RasterLayer` when `as_Spatial=TRUE`)

Author(s)

D.E. Beaudette and A.G. Brown

Examples

```
## Not run:
  library(terra)

  # soil order
  taxa <- 'vertisols'
  x <- taxaExtent(taxa, level = 'order')

  # suborder
  taxa <- 'ustalfs'
  x <- taxaExtent(taxa, level = 'suborder')

  # greatgroup
  taxa <- 'haplohumults'
  x <- taxaExtent(taxa, level = 'greatgroup')

  # subgroup
  taxa <- 'Typic Haploxerepts'
  x <- taxaExtent(taxa, level = 'subgroup')

  # greatgroup formative element
  taxa <- 'psamm'
  x <- taxaExtent(taxa, level = 'greatgroup', formativeElement = TRUE)

  # subgroup formative element
  taxa <- 'abruptic'
  x <- taxaExtent(taxa, level = 'subgroup', formativeElement = TRUE)
```

```
  # coarsen for faster plotting
  a <- terra::aggregate(x, fact = 5, na.rm = TRUE)

  # quick evaluation of the result
  terra::plot(a, axes = FALSE)

## End(Not run)
```

---

uncode *Convert coded values returned from NASIS and SDA queries into human-readable values*

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

```
uncode(
  df,
  invert = FALSE,
  db = "NASIS",
  droplevels = FALSE,
  stringsAsFactors = NULL,
  dsn = NULL
)

code(df, db = NULL, droplevels = FALSE, stringsAsFactors = NULL, dsn = NULL)
```

Arguments

df	data.frame
invert	converts the code labels back to their coded values (FALSE)
db	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")
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	deprecated
dsn	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.

Set `options(soilDB.NASIS.skip_uncode = TRUE)` to bypass decoding logic; for instance when using soilDB NASIS functions with custom NASIS snapshots that have already been decoded.

Value

A `data.frame` with the results.

Author(s)

Stephen Roecker

Examples

```
# convert column name `fraghard` (fragment hardness) codes to labels
uncode(data.frame(fraghard = 1:10))

# convert column name `fragshp` (fragment shape) labels to codes
code(data.frame(fragshp = c("flat", "nonflat")))
```

`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 *Compute Water Day and Year*

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", format = "%Y-%m-%d", tz = "UTC")
```

Arguments

d	anything the can be safely converted to POSIXlt
end	"MM-DD" notation for end of water year
format	Used in POSIXlt conversion. Default "%Y-%m-%d"
tz	Used in POSIXlt conversion for custom timezone. Default is "UTC"

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

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", "soilColor"))
```

Arguments

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

Value

a `data.frame`

Examples

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

Index

∗ **IO**
parseWebReport, 129
∗ **datasets**
loafercreek, 117
metadata, 122
NASIS_table_column_keys, 127
SCAN_SNOTEL_metadata, 133
soilDB.env, 146
us_ss_timeline, 158
∗ **hplot**
STRplot, 149
∗ **manip**
estimateSTR, 11
fetchNASISLabData, 21
fetchNASISWebReport, 22
fetchOSD, 24
fetchPedonPC, 26
fetchSCAN, 29
get_colors_from_NASIS_db, 46
get_colors_from_pedon_db, 47
get_comonth_from_NASIS_db, 48
get_component_data_from_NASIS_db, 49
get_component_from_GDB, 51
get_component_from_SDA, 53
get_cosoilmoist_from_NASIS, 55
get_extended_data_from_NASIS_db, 58
get_extended_data_from_pedon_db, 59
get_hz_data_from_NASIS_db, 60
get_hz_data_from_pedon_db, 61
get_lablayer_data_from_NASIS_db, 61
get_labpedon_data_from_NASIS_db, 62
get_site_data_from_NASIS_db, 102
get_site_data_from_pedon_db, 104
get_soilseries_from_NASIS, 104
get_text_notes_from_NASIS_db, 109
get_veg_data_from_NASIS_db, 110
get_veg_from_AK_Site, 111
get_veg_from_MT_veg_db, 111
get_veg_from_NPS_PLOTS_db, 112
get_veg_other_from_MT_veg_db, 113
get_veg_species_from_MT_veg_db, 113
OSDquery, 127
SDA_query, 133
SDA_spatialQuery, 135
siblings, 141
simplifyArtifactData, 143
simplifyColorData, 144
SoilWeb_spatial_query, 148
summarizeSoilTemperature, 150
uncode, 157
waterDayYear, 159
∗ **utilities**
fetchKSSL, 14

aqp::col2Munsell(), *10*, *145*
aqp::mixMunsell(), *11*

code *(*uncode*)*, 157
createSSURGO, 5
createSSURGO(), *10*
createStaticNASIS, 7

dbConnectNASIS, 8
dbQueryNASIS, 8
downloadSSURGO, 9
downloadSSURGO(), *6*

estimateColorMixture, 10
estimateSTR, 11, *150*

fetchGDB *(*get_component_from_GDB*)*, 51
fetchHenry *(*summarizeSoilTemperature*)*, 150
fetchHWSD, 12

fetchKSSL, 14
fetchLDM, 16
fetchNASIS, *4*, 19, *48*, *50*, *56*
fetchNASIS(), *71*, *103*
fetchNASISLabData, 21
fetchNASISWebReport, 22
fetchOSD, *15*, 24, *29*, *128*, *142*
fetchPedonPC, 26
fetchRaCA, 27
fetchSCAN, 29, *152*
fetchSDA *(*get_component_from_SDA*)*, 53
fetchSDA_spatial, 32
fetchSoilGrids, 34
fetchSOLUS, 37
fetchSRI, 40
fetchVegdata, 41
fetchVegdata(), *103*
filter_geochem, 44
format_SQL_in_statement, 45

get_chorizon_from_NASISWebReport *(*fetchNASISWebReport*)*, 22
get_chorizon_from_SDA *(*get_component_from_SDA*)*, 53
get_cointerp_from_SDA *(*get_component_from_SDA*)*, 53
get_colors_from_NASIS_db, 46
get_colors_from_pedon_db, 47, *61*
get_comonth_from_NASIS_db, 48
get_competing_soilseries_from_NASIS *(*get_soilseries_from_NASIS*)*, 104
get_component_cogeomorph_data_from_NASIS_db *(*get_component_data_from_NASIS_db*)*, 49
get_component_cogeomorph_data_from_NASIS_db2 *(*get_component_data_from_NASIS_db*)*, 49
get_component_copm_data_from_NASIS_db *(*get_component_data_from_NASIS_db*)*, 49
get_component_correlation_data_from_NASIS_db *(*get_component_data_from_NASIS_db*)*, 49
get_component_data_from_NASIS_db, 49
get_component_diaghz_from_NASIS_db *(*get_component_data_from_NASIS_db*)*, 49
get_component_esd_data_from_NASIS_db *(*get_component_data_from_NASIS_db*)*, 49
get_component_from_GDB, 51
get_component_from_NASISWebReport *(*fetchNASISWebReport*)*, 22
get_component_from_SDA, 53
get_component_horizon_data_from_NASIS_db *(*get_component_data_from_NASIS_db*)*, 49
get_component_otherveg_data_from_NASIS_db *(*get_component_data_from_NASIS_db*)*, 49
get_component_restrictions_from_NASIS_db *(*get_component_data_from_NASIS_db*)*, 49
get_concentrations_from_NASIS_db *(*fetchNASIS*)*, 19
get_copedon_from_NASIS_db *(*get_component_data_from_NASIS_db*)*, 49
get_cosoilmoist_from_NASIS, 55
get_cosoilmoist_from_NASISWebReport, *56*
get_cosoilmoist_from_NASISWebReport *(*fetchNASISWebReport*)*, 22
get_cosoilmoist_from_SDA, *56*
get_cosoilmoist_from_SDA *(*get_component_from_SDA*)*, 53
get_cotext_from_NASIS_db *(*get_text_notes_from_NASIS_db*)*, 109
get_ecosite_history_from_NASIS_db, 57
get_EDIT_ecoclass_by_geoUnit, 57
get_extended_data_from_NASIS_db, 58
get_extended_data_from_NASIS_db(), *57*
get_extended_data_from_pedon_db, 59
get_HWSD_path *(*fetchHWSD*)*, 12
get_hz_data_from_NASIS_db, *47*, *59*, *60*, 60
get_hz_data_from_NASIS_db(), *103*
get_hz_data_from_pedon_db, *27*, *47*, *60*, 61, *104*, *109*, *111*
get_lablayer_data_from_NASIS_db, 61, *63*
get_labpedon_data_from_NASIS_db, *21*, *62*, 62
get_legend_from_GDB *(*get_component_from_GDB*)*, 51
get_legend_from_NASIS

(get_mapunit_from_NASIS), 63
get_legend_from_NASISWebReport (fetchNASISWebReport), 22
get_legend_from_SDA (get_component_from_SDA), 53
get_lmuaoverlap_from_NASIS (get_mapunit_from_NASIS), 63
get_lmuaoverlap_from_NASISWebReport (fetchNASISWebReport), 22
get_lmuaoverlap_from_SDA (get_component_from_SDA), 53
get_mapunit_from_GDB (get_component_from_GDB), 51
get_mapunit_from_NASIS, 63
get_mapunit_from_NASISWebReport (fetchNASISWebReport), 22
get_mapunit_from_SDA (get_component_from_SDA), 53
get_mutext_from_NASIS_db (get_text_notes_from_NASIS_db), 109
get_NASIS_column_metadata (get_NASIS_metadata), 64
get_NASIS_fkey_by_name (get_NASIS_table_key_by_name), 65
get_NASIS_metadata, 64
get_NASIS_pkey_by_name (get_NASIS_table_key_by_name), 65
get_NASIS_pkeyref_by_name (get_NASIS_table_key_by_name), 65
get_NASIS_table_key_by_name, 65
get_NASIS_table_metadata, 66
get_NASIS_table_name_by_purpose, 67
get_NOAA_GHCND, 68
get_NOAA_stations_nearXY, 69
get_OSD, 70
get_OSD_JSON (get_OSD), 70
get_phfmp_from_NASIS_db (fetchNASIS), 19
get_phorizon_from_NASIS_db (fetchNASIS), 19
get_phroots_from_NASIS_db, 71
get_progress_from_NASISWebReport (fetchNASISWebReport), 22
get_project_correlation_from_NASISWebReport (fetchNASISWebReport), 22
get_project_from_NASISWebReport (fetchNASISWebReport), 22
get_projectmapunit2_from_NASISWebReport (fetchNASISWebReport), 22
get_projectmapunit_from_NASIS (get_projectmilestone_from_NASIS), 71
get_projectmapunit_from_NASISWebReport (fetchNASISWebReport), 22
get_projectmilestone_from_NASIS, 71
get_RMF_from_NASIS_db, 72
get_SDA_coecoclass, 73
get_SDA_cosurfmorph, 74
get_SDA_hydric, 76
get_SDA_interpretation, 78
get_SDA_metrics, 95
get_SDA_muaggatt, 96
get_SDA_pmgroupname, 97
get_SDA_property, 98
get_SDV_legend_elements, 101
get_site_association_from_NASIS (get_site_data_from_NASIS_db), 102
get_site_data_from_NASIS_db, *47*, *59, 60*, 102
get_site_data_from_pedon_db, *47*, *60, 61*, 104, *109*, *111*
get_sitesoilmoist_from_NASISWebReport (fetchNASISWebReport), 22
get_soilDB_env (soilDB.env), 146
get_soilseries_from_NASIS, 104
get_soilseries_from_NASISWebReport (get_soilseries_from_NASIS), 104
get_SRI, *40, 41*, 106, *108*
get_SRI_layers, *40*, *107*, 108
get_text_notes_from_NASIS_db, 109
get_veg_data_from_NASIS_db, 110
get_veg_from_AK_Site, *104*, 111
get_veg_from_MT_veg_db, 111, *113, 114*
get_veg_from_NPS_PLOTS_db, 112
get_veg_other_from_MT_veg_db, *112*, 113, *114*
get_veg_species_from_MT_veg_db, *112, 113*, 113
get_vegplot_from_NASIS_db (fetchVegdata), 41
get_vegplot_groundsurface_from_NASIS_db

(fetchVegdata), 41
get_vegplot_location_from_NASIS_db (fetchVegdata), 41
get_vegplot_prodquadrats_from_NASIS_db (fetchVegdata), 41
get_vegplot_species_from_NASIS_db (fetchVegdata), 41
get_vegplot_speciesbasalarea_from_NASIS (fetchVegdata), 41
get_vegplot_textnote_from_NASIS_db (fetchVegdata), 41
get_vegplot_transect_from_NASIS_db (fetchVegdata), 41
get_vegplot_transpecies_from_NASIS_db (fetchVegdata), 41
get_vegplot_transpoints_from_NASIS_db (fetchVegdata), 41
get_vegplot_tree_si_details_from_NASIS_db (fetchVegdata), 41
get_vegplot_tree_si_summary_from_NASIS_db (fetchVegdata), 41
get_vegplot_trhi_from_NASIS_db (fetchVegdata), 41
getHzErrorsNASIS, 45
getHzErrorsPedonPC, *27*
getHzErrorsPedonPC (fetchPedonPC), 26
gopheridge (loafercreek), 117

ISSR800.wcs, 114

KSSL_VG_model, 115

Lab Data Mart (LDM), *126*
loafercreek, *4*, 117
local database containing NASIS table structure, *20*, *64*, *66*, *71*, *72*, *105*, *125*
local_NASIS_defined, 118

make_EDIT_service_URL, 120
makeChunks, 119
makeChunks(), *25*
metadata, 122
mineralKing (loafercreek), 117
month2season (summarizeSoilTemperature), 150
mukey.wcs, 122

NASIS (dbConnectNASIS), 8
NASIS_table_column_keys, 127
NASISChoiceList, 124
NASISDomainsAsFactor, 125
NASISLocalDatabase, 126

OSDquery, 127, *142*
OSDquery(), *26*

parseWebReport, 129
processSDA_WKT, 130

ROSETTA, 131

SCAN_sensor_metadata (fetchSCAN), 29
SCAN_site_metadata (fetchSCAN), 29
SCAN_SNOTEL_metadata, 133
SDA_query, *4*, *55*, 133, *137*
SDA_spatialQuery, 135
SDA_spatialQuery(), *134*, *149*
seriesExtent, 139
siblings, *128*, 141, *142*
siblings(), *26*
simplifyArtifactData, 143
simplifyColorData, *47*, 144
simplifyFragmentData (simplifyArtifactData), 143
soilColor.wcs, 145
soilDB (soilDB-package), 4
soilDB-package, 4
soilDB.env, 146
soilDB_user_dir, 147
SoilWeb_spatial_query, 148
state_FIPS_codes (SCAN_SNOTEL_metadata), 133
STRplot, *12*, 149
summarizeSoilTemperature, 150

taxaExtent, 152

uncode, 157
us_ss_timeline, 158

waterDayYear, 159
WCS_details, 160