Package 'soilDB'

January 23, 2018

Type Package

Title Soil Database Interface

Version 2.0-1

Date 2018-01-23

Author D.E. Beaudette and J. Skovlin, and S.M. Roecker

Maintainer D.E. Beaudette <dylan.beaudette@ca.usda.gov>

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

License GPL (>= 3)

LazyLoad yes

Depends R (>= 3.0.0), aqp

Imports grDevices, graphics, stats, utils, plyr, Hmisc, xml2, sp, reshape2, raster, curl

Suggests lattice, dismo, rgdal, jsonlite, RODBC, httr, rgeos, rvest

Repository CRAN

URL http://ncss-tech.github.io/AQP/

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

RoxygenNote 6.0.1

NeedsCompilation no

Date/Publication 2018-01-23 21:47:02 UTC

R topics documented:

`soilDB-package` *Soil Database Interface*

Description

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

Details

It can be difficult to locate all of the dependencies required for sending/processing SOAP requests, especially on UNIX-like operating systems. Windows binary packages for the dependencies can be found here. See fetchPedonPC for a simple wrapper function that should suffice for typical site/pedon/hz queries. An introduction to the soilDB package can be found here.

Author(s)

J.M. Skovlin and D.E. Beaudette

See Also

fetchPedonPC, fetchNASIS, SDA_query, loafercreek

---

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

Description

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

Usage

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

Arguments

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

Details

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

Value

a list containing:

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

Note

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

Author(s)

D.E. Beaudette

See Also

`fetchSCAN`

Examples

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

# get CA630 data as daily averages
x <- fetchHenry(project='CA630', gran = 'day')

# inspect data gaps
levelplot(factor(!is.na(sensor_value)) ~ doy * factor(year) | name,
data=x$soiltemp, col.regions=c('grey', 'RoyalBlue'), cuts=1,
colorkey=FALSE, as.table=TRUE, scales=list(alternating=3),
par.strip.text=list(cex=0.75), strip=strip.custom(bg='yellow'),
xlab='Julian Day', ylab='Year')

## End(Not run)
```

fetchKSSL *Fetch KSSL Data (EXPERIMENTAL)*

Description

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

Usage

```
fetchKSSL(series=NULL, bbox=NULL, mlra=NULL, pedlabsampnum=NULL,
pedon_id=NULL, pedon_key=NULL, returnMorphologicData=FALSE)
```

Arguments

series	a single soil series name, case insensitive
bbox	a bounding box in WGS84 geographic coordinates e.g. `c(-120, 37, -122, 38)`
mlra	an MLRA ID, e.g. "18" or "22A"
pedlabsampnum	a single KSSL pedon lab sample number
pedon_id	a single user pedon ID
pedon_key	a single KSSL internal pedon ID
returnMorphologicData	optionally request basic morphologic data, see details section

Details

This is an experimental interface to a subset for the most commonly used data from a snapshot of KSSL (lab characterization) and NASIS (morphologic) data. The snapshots were last updated September 2017 (KSSL / NASIS).

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: horizon colors, rock fragments, pores, and structure. 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.

Function arguments (`series`, `mlra`, etc.) are NOT vectorized: the first element of a vector will be used when supplied as a filter. See the fetchKSSL tutorial for ideas on how to iterate over a set of IDs.)

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

References

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

See Also

fetchOSD

Examples

```
## Not run:
# 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
par(mar=c(0,0,0,0))
plot(s, name='hzn_desgn', max.depth=150)


# get morphologic data too
library(soilDB)
library(plyr)
library(reshape2)


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

# extract SPC
pedons <- s$SPC

# simplify color data
s.colors <- simplifyColorData(s$morph$phcolor, id.var = 'labsampnum')

# merge color data into SPC
h <- horizons(pedons)
h <- join(h, s.colors, by='labsampnum', type='left', match='first')
```

```
horizons(pedons) <- h

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


# simplify fragment data
s.frags <- simplfyFragmentData(s$morph$phfrags, id.var='labsampnum')

# merge fragment data into SPC
h <- horizons(pedons)
h <- join(h, s.frags, by='labsampnum', type='left', match='first')
horizons(pedons) <- h


# check
par(mar=c(0,0,3,0))
plot(pedons, color='total_frags_pct', print.id=FALSE)



## End(Not run)
```

`fetchLIMS_component` *Extract component tables from a the Laboratory Information Management System*

Description

Get, format, impute, and return component tables.

Usage

```
fetchLIMS_component(projectname, rmHzErrors = FALSE, fill = FALSE,
                    stringsAsFactors = default.stringsAsFactors()
                    )
get_progress_from_LIMS(mlrassoarea, fiscalyear, projecttypename)
get_project_from_LIMS(mlrassoarea, fiscalyear)
get_reverse_correlation_from_LIMS(mlrassoarea, fiscalyear, projectname)
get_mapunit_from_LIMS(projectname, stringsAsFactors = default.stringsAsFactors())
get_component_from_LIMS(projectname, stringsAsFactors = default.stringsAsFactors())
get_chorizon_from_LIMS(projectname, fill = FALSE,
                       stringsAsFactors = default.stringsAsFactors()
                       )
get_cosoilmoist_from_LIMS(projectname, impute = TRUE,
                          stringsAsFactors = default.stringsAsFactors()
                          )
get_sitesoilmoist_from_LIMS(usiteid)
```

Arguments

projectname	text string vector of project names to be inserted into a SQL WHERE clause (default: NA)
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)
usiteid	text string value identifying the user site id 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)
fill	should rows with missing component ids be removed NA (FALSE)
rmHzErrors	should pedons with horizonation errors be removed from the results? (default: FALSE)
stringsAsFactors	logical: should character vectors be converted to factors? This argument is passed to the uncode() function. It does not convert those vectors that have been set outside of uncode() (i.e. hard coded). The 'factory-fresh' default is TRUE, but this can be changed by setting options(stringsAsFactors = FALSE)

Value

A dataframe or list with the results.

Author(s)

Stephen Roecker

Examples

```
## Not run:
library(soilDB)
library(ggplot2)
library(gridExtra)

# query soil components by projectname
test = fetchLIMS_component(
"EVAL - MLRA 111A - Ross silt loam, 0 to 2 percent slopes, frequently flooded"
)
test = test$spc

# profile plot
plot(test)
```

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

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

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

h = merge(test2, site(test)[c("dmuiid", "coiid", "compname", "comppct_r")],
          by = "coiid",
          all.x = TRUE
          )

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

grid.arrange(g1, g2)


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

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

```
## End(Not run)
```

`fetchNASIS` *Fetch commonly used site/pedon/horizon or component data from a local NASIS database.*

Description

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

Usage

```
fetchNASIS(from = 'pedons', ...)

fetchNASIS_pedons(SS=TRUE, rmHzErrors=TRUE, nullFragsAreZero=TRUE,
                  soilColorState='moist', lab=FALSE,
                  stringsAsFactors = default.stringsAsFactors()
                  )
fetchNASIS_components(SS=TRUE, rmHzErrors=TRUE, fill = FALSE,
                      stringsAsFactors = default.stringsAsFactors()
                      )
getHzErrorsNASIS(strict=TRUE)
```

Arguments

`from`	determines what objects should fetched? ('pedons' | 'components')
`...`	arguments passed to `fetchNASIS_pedons()` or `fetchNASIS_components()`
`SS`	fetch data from the currently loaded selected set in NASIS or from the entire local database (default: TRUE)
`stringsAsFactors`	logical: should character vectors be converted to factors? This argument is passed to the uncode() function. It does not convert those vectors that have been set outside of uncode() (i.e. hard coded). The 'factory-fresh' default is TRUE, but this can be changed by setting options(stringsAsFactors = FALSE)
`rmHzErrors`	should pedons with horizonation errors be removed from the results? (default: TRUE)
`nullFragsAreZero`	should fragment volumes of NULL be interpreted as 0? (default: TRUE), see details
`soilColorState`	which colors should be used to generate the convenience field 'soil_color'? ('moist' | 'dry')
`lab`	should the phlabresults child table be fetched with site/pedon/horizon data (default: FALSE)

fill	should missing "month" rows in the comonth table be filled with NA (FALSE)
strict	how strict should horizon boundaries be checked for consistency: TRUE=more | FALSE=less

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 interpretated as NULLs. 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. Pedons that are missing horizon data, or have errors in their horizonation are excluded from the returned object, however, their IDs are printed on the console. Pedons with combination horizons (e.g. B/C) are erroneously marked as errors due to the way in which they are stored in NASIS as two overlapping horizon records.

See getHzErrorsNASIS for a simple approach to identifying pedons with problematic horizonation.

See the NASIS component tutorial, and NASIS pedon tutorial for more information.

Value

a SoilProfileCollection class object

Note

This function currently works only on Windows, and requires a 'nasis_local' ODBC connection.

Author(s)

D. E. Beaudette and J. M. Skovlin

Examples

```
## Not run:
# query depends on some pedon data, queried against the national database
# note that you must setup this connection ahead of time
f <- fetchNASIS(from = 'pedons')

# plot only those profiles with densic contact
plot(f[which(f$densic.contact), ], name='hzname')

# get basic component data from local NASIS, after performing a
# DMU-* query against the national database
fc <- fetchNASIS(from = 'components')

## End(Not run)
```

`fetchNASISLabData` *Fetch lab data used site/horizon data from a PedonPC database.*

Description

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

Usage

```
fetchNASISLabData()
```

Details

This function currently works only on Windows, and requires a 'nasis_local' ODBC connection.

Value

a SoilProfileCollection class object

Note

This fuction attempts to do most of the boilerplate work when extracting KSSL laboratory site/horizon data from a local NASIS database. Lab pedons that have errors in their horizonation are excluded from the returned object, however, their IDs are printed on the console. See `getHzErrorsNASIS` for a simple approach to identifying pedons with problematic horizonation.

Author(s)

J.M. Skovlin and D.E. Beaudette

See Also

`get_labpedon_data_from_NASIS_db`

Examples

```
## Not run:
# query depends on some lab data, queried against the national database
# note that you must setup this connection ahead of time
# see inst/doc/setup_ODBC_local_NASIS.pdf
f <- fetchNASISLabData()

# plot only those profiles with densic contact
#plot(f[which(f$densic.contact), ], name='hzname')


## End(Not run)
```

fetchOSD *Fetch Official Series Description (OSD) Data*

Description

This functions fetches a limited subset of horizon and site-level attributes for named soil series, from the SoilWeb system.

Usage

```
fetchOSD(soils, colorState = 'moist')
```

Arguments

soils	a character vector of named soil series
colorState	color state for horizon soil color visualization: "moist" or "dry"

Details

the search is case-insensitive

Value

a SoilProfileCollection object containing basic soil morphology and taxonomic information.

Note

SoilWeb maintains a snapshot of the Official Series Description data. Please use the link above for the live data.

Author(s)

D.E. Beaudette

References

http://www.nrcs.usda.gov/wps/portal/nrcs/detailfull/soils/home/?cid=nrcs142p2_053587

Examples

```
## Not run:
# 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))
plot(s.moist, name='hzname', cex.names=0.85, axis.line.offset=-4)
plot(s.dry, name='hzname', cex.names=0.85, axis.line.offset=-4)

## End(Not run)
```

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

Description

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

Usage

```
fetchPedonPC(dsn)
getHzErrorsPedonPC(dsn, strict=TRUE)
```

Arguments

`dsn`	The path to a PedonPC version 5.x database
`strict`	should horizonation by strictly enforced? (TRUE)

Details

This function currently works only on Windows.

Value

a SoilProfileCollection class object

Note

This fuction 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`

Examples

```
## Not run:
# path to local PedonPC back-end DB
dsn <- "S:/Service_Center/NRCS/pedon/pedon.accdb"

# get routinely used soil data SoilProfileCollection object
f <- fetchPedonPC(dsn)

# plot only those profiles with densic contact
plot(f[which(f$densic.contact), ], name='hzname')

## End(Not run)
```

`fetchRaCA` *Fetch KSSL Data (EXPERIMENTAL)*

Description

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

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`	an RaCA site id (e.g. 'C1609C01')
`get.vnir`	boolean, 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

http://www.nrcs.usda.gov/wps/portal/nrcs/detail/soils/survey/?cid=nrcs142p2_054164
fetchRaCA() Tutorial

See Also

fetchOSD

Examples

```
## Not run:
  # search by series name
  s <- fetchRaCA(series='auburn')

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

  # check structure
  str(s, 1)

  # extract pedons
  p <- s$pedons

  # how many pedons
  length(p)

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

## End(Not run)
```

---

fetchSCAN *Fetch SCAN Data*

Description

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

Usage

```
# get SCAN data
fetchSCAN(site.code, year, report='SCAN', req=NULL)
# get SCAN station metadata
SCAN_sensor_metadata(site.code)
```

Arguments

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

Details

See The fetchSCAN tutorial for details. These functions require the 'httr' and 'rvest' libraries.

Value

a data.frame object

Note

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

Author(s)

D.E. Beaudette

References

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

Examples

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

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

## End(Not run)
```

---

fetchSDA_component *Extract component tables from Soil Data Access*

Description

Get, format, impute, and return component tables.

Usage

```
fetchSDA_component(WHERE = NULL, duplicates = FALSE, rmHzErrors = FALSE,
                   stringsAsFactors = default.stringsAsFactors()
                   )
get_mapunit_from_SDA(WHERE = NULL, stringsAsFactors = default.stringsAsFactors())
get_component_from_SDA(WHERE = NULL, duplicates = FALSE,
                       stringsAsFactors = default.stringsAsFactors()
                       )
get_chorizon_from_SDA(WHERE = NULL, duplicates = FALSE,
                      stringsAsFactors = default.stringsAsFactors()
                      )
get_cosoilmoist_from_SDA(WHERE = NULL, duplicates = FALSE, impute = TRUE,
                         stringsAsFactors = default.stringsAsFactors()
                         )
get_cosoilmoist_from_NASIS(impute = TRUE, stringsAsFactors = default.stringsAsFactors())
```

Arguments

WHERE	text string formated as an SQL WHERE clause (default: FALSE)
duplicates	logical; if TRUE duplicate nationalmusym are returned
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)
rmHzErrors	should pedons with horizonation errors be removed from the results? (default: FALSE) UE)
stringsAsFactors	logical: should character vectors be converted to factors? This argument is passed to the uncode() function. It does not convert those vectors that have set outside of uncode() (i.e. hard coded). The 'factory-fresh' default is TRUE, but this can be changed by setting options(stringsAsFactors = FALSE)

Details

The SDA functions can get and fetch data with an internet connection and a WHERE clause.

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 function get_cosoilmoist_from_NASIS_db() only works only on Windows , and requires a 'nasis_local' ODBC connection. See the NASIS ODBC Setup tutorial for instructions.

Value

A dataframe or list with the results.

Author(s)

Stephen Roecker

Examples

```
## Not run:
library(soilDB)
library(ggplot2)
library(gridExtra)

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

# profile plot
plot(test)

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

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

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

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

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

grid.arrange(g1, g2)


# query cosoilmoist (e.g. water table data) by mukey
# NA depths are interpreted as (???) with impute=TRUE argument
x <- get_cosoilmoist_from_SDA(WHERE = "mukey = '1395352'", impute = TRUE)

ggplot(x, aes(x = as.integer(month), y = dept_r, lty = status)) +
  geom_rect(aes(xmin = as.integer(month), xmax = as.integer(month) + 1,
```

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


## End(Not run)
```

`get_colors_from_NASIS_db` *Extract Soil Color Data from a local NASIS Database*

Description

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

Usage

```
get_colors_from_NASIS_db(SS = TRUE)
```

Arguments

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

Details

This function currently works only on Windows.

Value

A dataframe with the results.

Author(s)

Jay M. Skovlin and Dylan E. Beaudette

See Also

simplifyColorData, get_hz_data_from_NASIS_db, get_site_data_from_NASIS_db

get_colors_from_pedon_db

Extract Soil Color Data from a PedonPC Database

Description

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

Usage

```
get_colors_from_pedon_db(dsn)
```

Arguments

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

Details

This function currently works only on Windows.

Value

A dataframe with the results.

Author(s)

Dylan E. Beaudette and Jay M. Skovlin

See Also

get_hz_data_from_pedon_db, get_site_data_from_pedon_db

get_comonth_from_NASIS_db

Extract component month data from a local NASIS Database

Description

Extract component month data from a local NASIS Database.

Usage

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

Arguments

SS	get data from the currently loaded Selected Set in NASIS or from the entire local database (default: TRUE)
fill	should missing "month" rows in the comonth table be filled with NA (FALSE)
stringsAsFactors	logical: should character vectors be converted to factors? This argument is passed to the uncode() function. It does not convert those vectors that have set outside of uncode() (i.e. hard coded). The 'factory-fresh' default is TRUE, but this can be changed by setting options(stringsAsFactors = FALSE)

Details

This function currently works only on Windows.

Value

A list with the results.

Author(s)

Stephen Roecker

See Also

fetchNASIS, fetchNASIS_components

Examples

```
## Not run:
# query text note data
cm <- get_comonth_from_NASIS_db()

# show structure of component month data
str(cm)


## End(Not run)
```

---

get_component_data_from_NASIS_db

Extract component data from a local NASIS Database

Description

Extract component data from a local NASIS Database.

Usage

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

Arguments

SS	get data from the currently loaded Selected Set in NASIS or from the entire local database (default: TRUE)
stringsAsFactors	logical: should character vectors be converted to factors? This argument is passed to the uncode() function. It does not convert those vectors that have set outside of uncode() (i.e. hard coded). The 'factory-fresh' default is TRUE, but this can be changed by setting options(stringsAsFactors = FALSE)

Details

This function currently works only on Windows.

Value

A list with the results.

Author(s)

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

See Also

fetchNASIS, fetchNASIS_components

Examples

```
## Not run:
# query text note data
fc <- get_component_data_from_NASIS_db()

# show structure of component data returned
str(fc)


## End(Not run)
```

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

Description

Extract accessory tables and summaries from a local NASIS Database.

Usage

```
get_extended_data_from_NASIS_db(SS = TRUE, nullFragsAreZero = TRUE,
                                stringsAsFactors = default.stringsAsFactors()
                                )
```

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

Details

This function currently works only on Windows.

Value

A list with the results.

Author(s)

Jay M. Skovlin and Dylan E. Beaudette

See Also

`get_hz_data_from_NASIS_db`, `get_site_data_from_NASIS_db`

Examples

```
## Not run:
# query extended data
e <- get_extended_data_from_NASIS_db()

# show contents of extended data
str(e)

## End(Not run)
```

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

Description

Extract accessory tables and summaries from a local pedonPC Database.

Usage

```
get_extended_data_from_pedon_db(dsn)
```

Arguments

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

Details

This function currently works only on Windows.

Value

A list with the results.

Author(s)

Jay M. Skovlin and Dylan E. Beaudette

See Also

`get_hz_data_from_pedon_db`, `get_site_data_from_pedon_db`

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

Description

Get horizon-level data from a local NASIS database.

Usage

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

Arguments

`SS`	fetch data from Selected Set in NASIS or from the entire local database (default: TRUE)
`stringsAsFactors`	logical: should character vectors be converted to factors? This argument is passed to the uncode() function. It does not convert those vectors that have been set outside of uncode() (i.e. hard coded). The 'factory-fresh' default is TRUE, but this can be changed by setting options(stringsAsFactors = FALSE)

Details

This function currently works only on Windows.

Value

A dataframe.

Note

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

Author(s)

Jay M. Skovlin and Dylan E. Beaudette

See Also

`get_hz_data_from_NASIS_db`, `get_site_data_from_NASIS_db`

get_hz_data_from_pedon_db *Extract Horizon Data from a PedonPC Database*

Description

Get horizon-level data from a PedonPC database.

Usage

```
get_hz_data_from_pedon_db(dsn)
```

Arguments

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

Details

This function currently works only on Windows.

Value

A dataframe.

Note

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

Author(s)

Dylan E. Beaudette and Jay M. Skovlin

See Also

get_colors_from_pedon_db, get_site_data_from_pedon_db

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

Description

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

Usage

```
get_lablayer_data_from_NASIS_db()
```

Details

This function currently works only on Windows, and requires a 'nasis_local' ODBC connection.

Value

A dataframe.

Note

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

Author(s)

Jay M. Skovlin and Dylan E. Beaudette

See Also

`get_labpedon_data_from_NASIS_db`

`get_labpedon_data_from_NASIS_db` *Extract lab pedon data from a local NASIS Database*

Description

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

Usage

```
get_labpedon_data_from_NASIS_db()
```

Details

This function currently works only on Windows, and requires a 'nasis_local' ODBC connection.

Value

A dataframe.

Note

This fuction 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_phlabresults_data_from_NASIS_db

Extract phlabresults table from a local NASIS Database

Description

Get, format, aggregate, and return phlabresults table from a NASIS database.

Usage

```
get_phlabresults_data_from_NASIS_db(SS=TRUE)
```

Arguments

SS	fetch data from the currently loaded selected set in NASIS or from the entire local database (default: TRUE)

Details

This function currently works only on Windows.

Value

A dataframe with the results.

Author(s)

Stephen Roecker

See Also

get_hz_data_from_NASIS_db, get_site_data_from_NASIS_db

`get_site_data_from_NASIS_db` *Extract Site Data from a local NASIS Database*

Description

Get site-level data from a local NASIS database.

Usage

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

Arguments

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

Details

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

Value

A dataframe.

Note

This function currently works only on Windows.

Author(s)

Jay M. Skovlin and Dylan E. Beaudette

See Also

get_hz_data_from_NASIS_db,

Examples

```
## Not run:

## Example: export / convert DMS coordinates from NASIS and save to DD import file

# load required libraries
library(soilDB)
library(rgdal)
library(plyr)

# get site data from NASIS
s <- get_site_data_from_NASIS_db()

# keep only those pedons with real coordinates
good.idx <- which(!is.na(s$x))
s <- s[good.idx, ]

# investigate multiple datums:
get_site_data_from_NASIS_db

## this is not universally appropriate!
# assume missing is NAD83
s$horizdatnm[is.na(s$horizdatnm)] <- 'NAD83'

# check: OK
table(s$horizdatnm, useNA='always')

# convert to NAD83
old.coords <- cbind(s$x, s$y)

# add temp column for projection information, and fill with proj4 style info
s$proj4 <- rep(NA, times=nrow(s))
s$proj4 <- paste('+proj=longlat +datum=', s$horizdatnm, sep='')

# iterate over pedons, and convert to WGS84
new.coords <- ddply(s, 'siteiid',
  .progress='text', .fun=function(i) {
    coordinates(i) <- ~ x + y
    proj4string(i) <- CRS(i$proj4)
    i.t <- spTransform(i, CRS('+proj=longlat +datum=WGS84'))
    i.c <- as.matrix(coordinates(i.t))
    return(data.frame(x.new=i.c[, 1], y.new=i.c[, 2]))
  })

# merge in new coordinates
s <- join(s, new.coords)

# any changes?
summary(sqrt(apply((s[, c('x', 'y')] - s[, c('x.new', 'y.new')])^2, 1, sum)))

# save to update file for use with "Import of Standard WGS84 Georeference" calculation in NASIS
```

```
# note that this defines the coordinate source as "GPS", hence the last column of '1's.
std.coordinates.update.data <- unique(cbind(s[, c('siteiid', 'y.new', 'x.new')], 1))
# save to file
write.table(std.coordinates.update.data,
file='c:/data/sgeoref.txt', col.names=FALSE, row.names=FALSE, sep='|')

## End(Not run)
```

`get_site_data_from_pedon_db` *Extract Site Data from a PedonPC Database*

Description

Get site-level data from a PedonPC database.

Usage

```
get_site_data_from_pedon_db(dsn)
```

Arguments

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

Value

A dataframe.

Note

This function currently works only on Windows.

Author(s)

Dylan E. Beaudette and Jay M. Skovlin

See Also

`get_hz_data_from_pedon_db`, `get_veg_from_AK_Site`,

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

Description

Extract text note data from a local NASIS Database.

Usage

```
get_text_notes_from_NASIS_db(SS = TRUE)
```

Arguments

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

Details

This function currently works only on Windows.

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

```
## Not run:
# query text note data
t <- 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
table(t$site_text$textcat)


## End(Not run)
```

`get_veg_data_from_NASIS_db` *Extract veg data from a local NASIS Database*

Description

Extract veg data from a local NASIS Database.

Usage

```
get_veg_data_from_NASIS_db(SS = TRUE)
```

Arguments

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

Details

This function currently works only on Windows.

Value

A list with the results.

Author(s)

Jay M. Skovlin and Dylan E. Beaudette

Examples

```
## Not run:
# query text note data
v <- get_veg_from_NASIS_db()

# show contents veg data returned
str(v)



## End(Not run)
```

get_veg_from_AK_Site *Retrieve Vegetation Data from an AK Site Database*

Description

Retrieve Vegetation Data from an AK Site Database

Usage

```
get_veg_from_AK_Site(dsn)
```

Arguments

dsn	file path the the AK Site access database

Value

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

Note

This function currently works only on Windows.

Author(s)

Dylan E. Beaudette

See Also

get_hz_data_from_pedon_db, get_site_data_from_pedon_db

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

Description

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

Usage

```
get_veg_from_MT_veg_db(dsn)
```

Arguments

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

Details

This function currently works only on Windows.

Value

A dataframe.

Author(s)

Jay M. Skovlin

See Also

get_veg_species_from_MT_veg_db, get_veg_other_from_MT_veg_db

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

Description

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

Usage

```
get_veg_from_NPS_PLOTS_db(dsn)
```

Arguments

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

Value

A dataframe with vegetation data in a long format with linkage to NRCS soil pedon data via the site_id key field.

Note

This function currently only works on Windows.

Author(s)

Jay M. Skovlin

get_veg_other_from_MT_veg_db

Extract cover composition data from a Montana RangeDB database

Description

Get cover composition data from a Montana RangeDB database.

Usage

```
get_veg_other_from_MT_veg_db(dsn)
```

Arguments

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

Details

This function currently works only on Windows.

Value

A dataframe.

Author(s)

Jay M. Skovlin

See Also

get_veg_from_MT_veg_db, get_veg_species_from_MT_veg_db

get_veg_species_from_MT_veg_db

Extract species-level Data from a Montana RangeDB database

Description

Get species-level data from a Montana RangeDB database.

Usage

```
get_veg_species_from_MT_veg_db(dsn)
```

Arguments

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

Details

This function currently works only on Windows.

Value

A dataframe.

Author(s)

Jay M. Skovlin

See Also

get_veg_from_MT_veg_db, get_veg_other_from_MT_veg_db

---

gSSURGO.chunk *Gridded SSURGO Chunk*

Description

A chunk of the gridded SSURGO database (gSSURGO)

Usage

```
data(gSSURGO.chunk)
```

Details

This is a 106x137 grid, cropped from the gSSURGO database. Cell values are map unit keys (mukey), stored as integers. No raster attribute table (RAT) is included with this sample data set. Note that this sample of the gSSURGO data has been modified such that cell values are map unit keys, rather than the gSSURGO integer key.

Source

http://www.nrcs.usda.gov/wps/portal/nrcs/detail//?cid=nrcs142p2_053628

Examples

```
## Not run:
data(gSSURGO.chunk)

## 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	a 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-tranformed values with units of cm

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

Value

A list with the following components:

VG_curve estimated water retention curve: paired estimates of water potential and water content

VG_inverse_function function for converting measured 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

van Genuchten, M.Th. (1980). "A closed-form equation for predicting the hydraulic conductivity of unsaturated soils". Soil Science Society of America Journal. 44 (5): 892-898.

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() as SoilProfileCollection objects.

Usage

```
data(loafercreek)
data(gopheridge)
```

Format

```
Formal class 'SoilProfileCollection' [package "aqp"] with 7 slots
  ..@ idcol     : chr "peiid"
  ..@ depthcols : chr [1:2] "hzdept" "hzdepb"
  ..@ metadata  :'data.frame': 1 obs. of  1 variable:
  .. ..$ depth_units: chr "cm"
  ..@ horizons  :'data.frame': 356 obs. of  54 variables:
  .. ..$ peiid            : int [1:356] 207252 207252 207252 207252 207252 242808 242808 242808 242808
  .. ..$ phiid            : int [1:356] 1018157 1018156 1018155 1018154 1018153 1148313 1148312 114831
  .. ..$ hzname               : chr [1:356] "A" "Bt1" "Bt2" "Bt3" ...
  .. ..$ genhz                : chr [1:356] "A" "Bt1" "Bt2" "Bt2" ...
  .. ..$ hzdept               : int [1:356] 0 18 41 61 81 0 3 5 10 18 ...
  .. ..$ hzdepb               : num [1:356] 18 41 61 81 152 3 5 10 18 36 ...
  .. ..$ clay                 : num [1:356] 34 32 34 32 NA ...
  .. ..$ silt                 : num [1:356] 21 28 31 33 NA ...
  .. ..$ sand                 : num [1:356] 45 40 35 35 NA ...
```

```
.. ..$ fragvoltot          : num [1:356] 0 0 0 0 0 0 0 6 11 63 ...
.. ..$ texture             : chr [1:356] "CBV-CL" "CB-CL" "CB-CL" "CBX-CL" ...
.. ..$ texcl             : Factor w/ 21 levels "cos","s","fs",..: 17 17 17 17 NA NA NA 13 13 13 ...
.. ..$ lieutex           : Factor w/ 58 levels "ashy","apum",..: NA NA NA NA NA 44 45 NA NA NA ...
.. ..$ phfield             : num [1:356] 6.4 6.5 6.5 6.7 NA ...
.. ..$ effclass          : Factor w/ 5 levels "very slight",..: 5 5 5 5 NA NA NA 5 5 5 ...
.. ..$ labsampnum          : chr [1:356] NA NA NA NA ...
.. ..$ rupresblkdry      : Factor w/ 11 levels "loose","soft",..: 5 6 6 6 NA NA NA 5 6 7 ...
.. ..$ stickiness        : Factor w/ 4 levels "moderately sticky",..: 3 1 1 1 NA NA NA 3 3 1 ...
.. ..$ plasticity        : Factor w/ 4 levels "moderately plastic",..: 1 1 1 1 NA NA NA 2 3 1 ...
.. ..$ texture_class       : chr [1:356] "cl" "cl" "cl" "cl" ...
.. ..$ d_r                 : num [1:356] 0.595 0.503 0.503 0.522 NA ...
.. ..$ d_g                 : num [1:356] 0.455 0.346 0.346 0.367 NA ...
.. ..$ d_b                 : num [1:356] 0.333 0.25 0.25 0.267 NA ...
.. ..$ d_hue               : chr [1:356] "7.5YR" "5YR" "5YR" "5YR" ...
.. ..$ d_value             : num [1:356] 5 4 4 4 NA NA NA 4 4 4 ...
.. ..$ d_chroma            : num [1:356] 4 4 4 4 NA NA NA 4 6 6 ...
.. ..$ d_sigma             : num [1:356] NA NA NA 0.0144 NA ...
.. ..$ m_r                 : num [1:356] 0.383 0.394 0.541 0.541 NA ...
.. ..$ m_g                 : num [1:356] 0.257 0.25 0.331 0.331 NA ...
.. ..$ m_b                 : num [1:356] 0.151 0.167 0.185 0.185 NA ...
.. ..$ m_hue               : chr [1:356] "7.5YR" "5YR" "5YR" "5YR" ...
.. ..$ m_value             : num [1:356] 3 3 4 4 NA NA NA 3 3 3 ...
.. ..$ m_chroma            : num [1:356] 4 4 6 6 NA NA NA 4 4 4 ...
.. ..$ m_sigma             : num [1:356] NA NA NA NA NA NA NA NA NA NA ...
.. ..$ moist_soil_color    : chr [1:356] "#624126" "#64402B" "#8A542F" "#8A542F" ...
.. ..$ dry_soil_color      : chr [1:356] "#987455" "#805840" "#805840" "#855D44" ...
.. ..$ soil_color          : chr [1:356] "#624126" "#64402B" "#8A542F" "#8A542F" ...
.. ..$ fine_gravel         : num [1:356] 0 0 0 0 0 0 0 0 0 0 ...
.. ..$ gravel              : num [1:356] 20 15 10 35 0 0 0 6 11 47 ...
.. ..$ cobbles             : num [1:356] 20 10 10 50 0 0 0 0 0 11 ...
.. ..$ stones              : num [1:356] 0 0 0 0 0 0 0 0 0 5 ...
.. ..$ boulders            : num [1:356] 0 0 0 0 0 0 0 0 0 0 ...
.. ..$ channers            : num [1:356] 0 0 0 0 0 0 0 0 0 0 ...
.. ..$ flagstones          : num [1:356] 0 0 0 0 0 0 0 0 0 0 ...
.. ..$ parafine_gravel     : num [1:356] 0 0 0 0 0 0 0 0 0 0 ...
.. ..$ paragravel          : num [1:356] 0 0 0 0 0 0 0 0 0 0 ...
.. ..$ paracobbles         : num [1:356] 0 0 0 0 0 0 0 0 0 0 ...
.. ..$ parastones          : num [1:356] 0 0 0 0 0 0 0 0 0 0 ...
.. ..$ paraboulders        : num [1:356] 0 0 0 0 0 0 0 0 0 0 ...
.. ..$ parachanners        : num [1:356] 0 0 0 0 0 0 0 0 0 0 ...
.. ..$ paraflagstones      : num [1:356] 0 0 0 0 0 0 0 0 0 0 ...
.. ..$ unspecified         : num [1:356] 0 0 0 0 0 0 0 0 0 0 ...
.. ..$ total_frags_pct_nopf: num [1:356] 40 25 20 85 0 0 0 6 11 63 ...
.. ..$ total_frags_pct     : num [1:356] 40 25 20 85 0 0 0 6 11 63 ...
..@ site      :'data.frame': 60 obs. of  83 variables:
.. ..$ peiid             : chr [1:60] "207252" "242808" "252851" "268791" ...
.. ..$ pedon_id          : chr [1:60] "06JCR005" "S2007CA009002" "S2007CA009002" "07JCR003" ...
```

```
.. ..$ siteiid          : int [1:60] 209257 254491 254491 269602 269603 269604 269605 269606 307646
.. ..$ site_id          : chr [1:60] "06CA630JCR005" "07-JCR-002" "07-JCR-002" "07CA630JCR003" ...
.. ..$ obs_date         : POSIXct[1:60], format: "2006-08-21" "2007-04-02" "2007-04-02" "2007-05-14
.. ..$ utmzone             : int [1:60] 10 10 10 10 10 10 10 10 10 10 ...
.. ..$ utmeasting          : num [1:60] 714807 700783 700783 714553 700765 ...
.. ..$ utmnorthing       : num [1:60] 4195263 4201935 4201935 4191747 4201934 ...
.. ..$ x                   : num [1:60] -121 -121 -121 -121 -121 ...
.. ..$ y                   : num [1:60] 37.9 37.9 37.9 37.8 37.9 ...
.. ..$ horizdatnm        : Factor w/ 25 levels "NAD27","NAD83",..: 2 2 2 2 2 2 2 2 2 2 ...
.. ..$ x_std               : num [1:60] -121 -121 -121 -121 -121 ...
.. ..$ y_std               : num [1:60] 37.9 37.9 37.9 37.8 37.9 ...
.. ..$ gpspositionalerror  : num [1:60] NA NA NA NA NA NA NA NA NA NA ...
.. ..$ describer         : chr [1:60] "John Rule" "John Rule" "John Rule" "John Rule" ...
.. ..$ pedonpurpose      : Factor w/ 10 levels "crop yield data site",..: 4 8 4 4 4 4 4 4 2 4 ...
.. ..$ pedontype         : Factor w/ 9 levels "map unit inclusion",..: 5 7 7 6 6 6 6 8 8 7 ...
.. ..$ pedlabsampnum       : chr [1:60] NA "07N0469" "07N0469" NA ...
.. ..$ labdatadescflag     : int [1:60] 0 1 0 0 0 0 0 0 0 0 ...
.. ..$ elev_field        : num [1:60] 213 337 337 324 323 352 412 314 149 336 ...
.. ..$ slope_field         : num [1:60] 47 20 20 15 20 38 31 22 NA 57 ...
.. ..$ aspect_field       : int [1:60] 68 272 272 154 272 340 262 82 NA 135 ...
.. ..$ plantassocnm      : chr [1:60] "Blue Oak Woodland" NA NA "Blue Oak Woodland" ...
.. ..$ earthcovkind1     : Factor w/ 10 levels "artificial cover",..: NA NA NA NA NA NA NA NA NA NA .
.. ..$ earthcovkind2     : Factor w/ 28 levels "row crop","close-grown crop",..: NA NA NA NA NA NA N
.. ..$ erocl            : Factor w/ 5 levels "0","1","2","3",..: NA NA NA NA NA NA NA NA NA NA ...
.. ..$ bedrckdepth         : int [1:60] 81 81 81 60 NA 91 NA 61 NA 91 ...
.. ..$ bedrckkind        : Factor w/ 157 levels "sandstone, unspecified",..: 104 104 104 106 104 104
.. ..$ bedrckhardness     : Factor w/ 14 levels "noncemented",..: 2 11 11 11 NA NA NA 11 NA 2 ...
.. ..$ hillslopeprof      : Factor w/ 5 levels "summit","shoulder",..: 3 3 3 3 3 3 NA NA NA 3 ...
.. ..$ geomslopeseg       : Factor w/ 9 levels "depression","drainageway",..: NA 4 4 NA NA NA NA NA NA
.. ..$ shapeacross        : Factor w/ 5 levels "concave","linear",..: 3 2 2 3 2 2 NA 2 NA 3 ...
.. ..$ shapedown         : Factor w/ 5 levels "concave","linear",..: 2 2 2 3 2 2 NA 2 NA 3 ...
.. ..$ slopecomplex       : Factor w/ 2 levels "complex","simple": 2 1 1 NA 1 NA NA NA NA 2 ...
.. ..$ drainagecl         : Factor w/ 8 levels "excessively",..: 3 3 3 3 3 3 3 3 NA 3 ...
.. ..$ geomposhill        : Factor w/ 8 levels "interfluve","head slope",..: NA 4 4 4 4 4 NA 4 NA 4 ..
.. ..$ geomposmntn        : Factor w/ 7 levels "mountaintop",..: NA NA NA NA NA NA NA NA NA NA ...
.. ..$ geomposflats       : Factor w/ 4 levels "flat","dip","rise",..: NA NA NA NA NA NA NA NA NA NA .
.. ..$ slope_shape        : Factor w/ 9 levels "linear / convex",..: 5 2 2 4 2 2 NA 2 NA 4 ...
.. ..$ classdate          : POSIXct[1:60], format: "2017-03-27 00:00:00" "2012-01-24 00:00:00" "2007-
.. ..$ classifier            : chr [1:60] NA NA NA NA ...
.. ..$ classtype          : Factor w/ 4 levels "correlated","field",..: 1 1 4 1 1 1 1 1 1 1 ...
.. ..$ taxonname          : chr [1:60] "Gopheridge" "Gopheridge" "Gopheridge" "Gopheridge" ...
.. ..$ localphase            : chr [1:60] NA NA NA NA ...
.. ..$ taxonkind          : Factor w/ 6 levels "family","taxon above family",..: 4 4 NA 4 4 4 4 4 5 4 .
.. ..$ seriesstatus       : Factor w/ 3 levels "tentative","established",..: NA 1 NA NA NA NA NA NA 2
.. ..$ taxpartsize        : Factor w/ 113 levels "unclassified",..: 33 33 33 33 33 33 33 33 63 33 ...
.. ..$ taxorder          : Factor w/ 12 levels "alfisols","andisols",..: 1 1 1 1 1 1 1 1 1 1 ...
.. ..$ taxsuborder        : Factor w/ 81 levels "aqualfs","boralfs",..: 5 5 5 5 5 5 5 5 5 5 ...
.. ..$ taxgrtgroup        : Factor w/ 459 levels "albaqualfs","duraqualfs",..: 38 38 38 38 38 38 38 38
```

```
.. ..$ taxsubgrp         : Factor w/ 3764 levels "typic albaqualfs",..: 270 270 266 270 270 270 270 2
.. ..$ soiltaxedition     : Factor w/ 12 levels "first edition",..: 12 10 10 10 12 10 10 12 12 12 ...
.. ..$ osdtypelocflag      : int [1:60] 0 1 0 0 0 0 0 0 0 0 ...
.. ..$ taxmoistcl        : Factor w/ 7 levels "aquic","aridic (torric)",..: 7 7 7 7 7 7 7 7 7 7 ...
.. ..$ taxtempregime      : Factor w/ 13 levels "cryic","frigid",..: 10 10 10 10 10 10 10 10 10 10 ...
.. ..$ taxfamother       : Factor w/ 13 levels "not used","coated",..: NA NA NA NA NA NA NA NA 6 NA .
.. ..$ psctopdepth         : int [1:60] NA 18 18 15 10 10 5 15 30 23 ...
.. ..$ pscbotdepth         : int [1:60] NA 68 68 58 60 60 55 61 45 73 ...
.. ..$ selection_method   : chr [1:60] "most recent" "most recent" "single record" "most recent" ...
.. ..$ ecositeid           : chr [1:60] NA NA NA NA ...
.. ..$ ecositenm           : chr [1:60] NA NA NA NA ...
.. ..$ ecositecorrdate     : POSIXct[1:60], format: NA NA NA NA ...
.. ..$ es_classifier       : chr [1:60] NA NA NA NA ...
.. ..$ es_selection_method : chr [1:60] NA NA NA NA ...
.. ..$ ochric.epipedon     : logi [1:60] TRUE TRUE TRUE TRUE TRUE TRUE ...
.. ..$ argillic.horizon    : logi [1:60] TRUE TRUE TRUE TRUE TRUE TRUE ...
.. ..$ lithic.contact      : logi [1:60] TRUE TRUE FALSE TRUE TRUE TRUE ...
.. ..$ cambic.horizon     : logi [1:60] FALSE FALSE FALSE FALSE FALSE FALSE ...
.. ..$ paralithic.contact  : logi [1:60] FALSE FALSE FALSE FALSE FALSE FALSE ...
.. ..$ mollic.epipedon    : logi [1:60] FALSE FALSE FALSE FALSE FALSE FALSE ...
.. ..$ paralithic.materials: logi [1:60] FALSE FALSE FALSE FALSE FALSE FALSE ...
.. ..$ surface_fgravel     : num [1:60] 0 5 5 0 0 0 0 0 0 0 ...
.. ..$ surface_gravel      : num [1:60] 0 11 11 0 5 5 0 5 0 5 ...
.. ..$ surface_cobbles     : num [1:60] 0 1 1 0 0 0 0 0 0 3 ...
.. ..$ surface_stones      : num [1:60] 0 0 0 0 0 0 0 0 0 2 ...
.. ..$ surface_boulders    : num [1:60] 0 0 0 0 0 0 0 0 0 0 ...
.. ..$ surface_channers    : num [1:60] 0 0 0 0 0 0 0 0 0 0 ...
.. ..$ surface_flagstones  : num [1:60] 0 0 0 0 0 0 0 0 0 0 ...
.. ..$ surface_paragravel  : num [1:60] 0 0 0 0 0 0 0 0 0 0 ...
.. ..$ surface_paracobbles : num [1:60] 0 0 0 0 0 0 0 0 0 0 ...
.. ..$ landform_string    : chr [1:60] "canyon" "hillslope" "hillslope" "hillslope" ...
.. ..$ pmkind            : chr [1:60] NA "colluvium & residuum" "colluvium & residuum" "colluvium & r
.. ..$ pmorigin          : chr [1:60] NA "greenstone" "greenstone" "metavolcanics" ...
..@ sp        :Formal class 'SpatialPoints' [package "sp"] with 3 slots
.. .. ..@ coords     : num [1, 1] 0
.. .. ..@ bbox       : logi [1, 1] NA
.. .. ..@ proj4string:Formal class 'CRS' [package "sp"] with 1 slot
.. .. .. .. ..@ projargs: chr NA
..@ diagnostic:'data.frame': 192 obs. of  4 variables:
.. ..$ peiid   : int [1:192] 207252 207252 207252 242808 242808 242808 252851 252851 268791 268791 ..
.. ..$ featkind: Factor w/ 84 levels "anthropic epipedon",..: 23 32 18 23 32 18 23 32 23 32 ...
.. ..$ featdept: int [1:192] 0 18 81 0 18 81 5 18 0 15 ...
.. ..$ featdepb: int [1:192] 18 81 152 18 81 NA 18 68 15 58 ...
```

Examples

```
## Not run:
# load example dataset
data("gopheridge")
```

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

# how many profiles?
length(gopheridge)

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

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

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

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

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

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

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

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

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

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

## End(Not run)
```

---

mapunit_geom_by_ll_bbox

Fetch Map Unit Geometry from SDA

Description

Fetch map unit geometry from the SDA website by WGS84 bounding box.

Usage

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

Arguments

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

Details

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

Value

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

Note

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

Author(s)

Dylan E Beaudette

References

http://casoilresource.lawr.ucdavis.edu/

Examples

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

## Not run:
# basic usage
b <- c(-120.54,38.61,-120.41,38.70)
x <- mapunit_geom_by_ll_bbox(b) # about 20 seconds

# note that the returned geometry is everything overlapping the bbox
```

```
# and not an intersection... why?
plot(x)
rect(b[1], b[2], b[3], b[4], border='red', lwd=2)


# get map unit data for matching map unit keys
in.statement <- format_SQL_in_statement(unique(x$MUKEY))
q <- paste("SELECT mukey, muname FROM mapunit WHERE mukey IN ", in.statement, sep="")
res <- SDA_query(q)

## End(Not run)
```

`parseWebReport` *Parse contents of a web report, based on suplied arguments.*

Description

Parse contents of a web report, based on suplied 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 specifiying the table to rerturn, or, NULL for a list of tables

Details

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

Value

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

Note

Most web reports are for internal use only.

Author(s)

D.E. Beaudette and S.M. Roecker

Examples

```
# pending
```

SDA_query *Soil Data Access Query*

Description

Submit a query to the Soil Data Acccess (SDA) website in SQL, get the results as a dataframe.

Usage

```
SDA_query(q)
```

Arguments

q	a valid T-SQL query surrounded by double quotes

Details

The SDA website can be found at http://sdmdataaccess.nrcs.usda.gov and query examples can be found at http://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 dataframe containing the results. NULL is retutned when queries result in 0 matches rows.

Note

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

Author(s)

D.E. Beaudette

See Also

mapunit_geom_by_ll_bbox

Examples

```
## Not run:


## get SSURGO export date for all soil survey areas in California
# there is no need to filter STATSGO
# because we are filtering on SSURGO areasymbols
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
-- critical: filter out STATSGO data
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
#
# requires raster and rgeos packages because
library(raster) # suggested by soilDB
library(rgeos)  # additional

# text -> bbox -> WKT
```

```
# xmin, xmax, ymin, ymax
b <- c(-120.9, -120.8, 37.7, 37.8)
p <- writeWKT(as(extent(b), 'SpatialPolygons'))
q <- paste0("SELECT mukey, cokey, compname, 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)

## End(Not run)
```

SDA_query_features *Soil Data Access Spatial Query*

Description

Iterate over Spatial* object features and submit spatial queries to the SDA web-service.

Usage

```
SDA_query_features(x, id='pedon_id')
processSDA_WKT(d, g='geom', p4s='+proj=longlat +datum=WGS84')
```

Arguments

x	a Spatial* object with more than 1 feature, any defined coordinate system
id	the column name in x that contains a unique ID for each feature
d	data.frame returned by SDA_query, containing WKT representation of geometry
g	name of column in d containing WKT geometry
p4s	PROJ4 CRS defs

Details

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

Value

A dataframe containing the results.

Note

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

Author(s)

D.E. Beaudette

`seriesExtent` *Get/Display Soil Series Extent*

Description

Get or display the spatial extent of a named soil series using the Series Extent Explorer.

Usage

```
seriesExtent(s, timeout=60)
seriesExtentAsGmap(s, timeout=60, exp=1.25)
```

Arguments

`s`	the soil series name
`timeout`	time that we are willing to wait for a response, in seconds
`exp`	expansion factor used to expand Google Maps region

Details

Soil series extent data are downloaded from a static cache of GeoJSON files on SoilWeb servers. Cached data are typically updated annually.

Value

when calling `seriesExtent`, a SpatialPolygonsDataFrame object

Note

These function require the 'rgdal' and 'dismo' packages.

Author(s)

D.E. Beaudette

References

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

Examples

```
## Not run:
# fetch series extent for the 'Amador' soil series
s <- seriesExtent('amador')
plot(s)

# fetch then plot the extent of the 'Amador' soil series
seriesExtentAsGmap('amador')

## End(Not run)
```

`simplfyFragmentData` *Simplify Coarse Fraction Data*

Description

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

Usage

```
simplfyFragmentData(rf, id.var, nullFragsAreZero = TRUE)
```

Arguments

`rf`	a `data.frame` object, typically returned from NASIS, see details
`id.var`	character vector with the name of the column containing an ID that is unique among all horizons in `rf`
`nullFragsAreZero`	should fragment volumes of NULL be interpreted as 0? (default: TRUE), see details

Details

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

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

There are examples in the KSSL data tutorial.

Author(s)

D.E. Beaudette

`simplifyColorData` *Simplify Color Data by ID*

Description

Simplify multiple Munsell color observations associated with each horizon.

Usage

```
simplifyColorData(d, id.var = "phiid", ...)
mix_and_clean_colors(x, wt='pct', colorSpace='LAB')
```

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`
`...`	further arguments passed on to `mix_and_clean_colors()`, see details
`x`	a `data.frame` object containing sRGB cordinates associated with a group of colors to mix
`wt`	a character vector with the name of the column containing color weights for mixing
`colorSpace`	a character vector with the name of color space in which mixing is performed ("LAB" or "sRGB")

Details

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

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".

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

There are examples in the KSSL data tutorial and the soil color mixing tutorial.

Author(s)

D.E. Beaudette

SoilWeb_spatial_query *Get SSURGO Data via Spatial Query*

Description

Get SSURGO Data via Spatial Query to SoilWeb

Usage

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

Arguments

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

Details

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

Value

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

Note

This function should be considered experimental; arguments, results, and side-effects could change at any time. SDA now supports spatial queries, consider using SDA_query_features instead.

Author(s)

D.E. Beaudette

Examples

```
# query by bbox
## Not run: SoilWeb_spatial_query(bbox=c(-122.05, 37, -122, 37.05))

# query by coordinate pair
## Not run: SoilWeb_spatial_query(coords=c(-121, 38))
```

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

Description

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

Usage

```
uncode(df, invert = FALSE, db = "NASIS", stringsAsFactors = default.stringsAsFactors())
code(df, ...)
```

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")
stringsAsFactors	logical: should character vectors be converted to factors? The 'factory-fresh' default is TRUE, but this can be changed by setting options(stringsAsFactors = FALSE)
...	arguments passed on to uncode

Details

These functions convert the coded values returned from NASIS into their plaintext representation. The converted values from NASIS, or sourced from SDA, are upgraded to specifically-leveled factors using the metadata tables from NASIS. For SDA the metadata is pulled from a static snapshot in the soilDB package.

Value

A dataframe with the results.

Author(s)

Stephen Roecker

Examples

```
## Not run:
# query component by nationalmusym
comp = fetchSDA_component(WHERE = "nationalmusym = '2vzcp'")
s = site(comp$spc)
s = uncode(s, NASIS = FALSE)
levels(s$taxorder)

## End(Not run)
```

Index