

Package ‘sharpshootR’

December 5, 2015

Type Package

Title A Soil Survey Toolkit

Description Miscellaneous soil data management, summary, visualization, and conversion utilities to support soil survey.

Version 0.9.1

Date 2015-12-04

Author USDA-NRCS Soil Survey Staff

Maintainer Dylan Beaudette <dylan.beaudette@ca.usda.gov>

License GPL (>= 2)

Repository CRAN

URL <http://aqp.r-forge.r-project.org/>

Suggests MASS, rgdal, spdep, circlize

Depends R (>= 3.0.0)

Imports grDevices, graphics, methods, stats, utils, aqp, ape, soilDB, igraph, cluster, lattice, latticeExtra, vegan, sp, reshape2, Hmisc, scales, circular, RColorBrewer, plyr, digest

Repository/R-Forge/Project aqp

Repository/R-Forge/Revision 1196

Repository/R-Forge/DateTimeStamp 2015-12-04 22:29:37

Date/Publication 2015-12-05 15:28:06

NeedsCompilation no

R topics documented:

sharpshootR-package	2
aggregateColorPlot	2
amador	3
aspect.plot	4
CDEC.snow.courses	5
CDECquery	6

CDECsnowQuery	9
component.adj.matrix	10
diagnosticPropertyPlot	11
dist.along.grad	13
dueling.dendrograms	14
generateLineHash	15
hillslope.probability	16
joinAdjacency	17
multinomial2logical	18
plotAvailWater	19
plotSoilRelationGraph	20
plotTransect	22
polygonAdjacency	24
sample.by.poly	25
SoilTaxonomyDendrogram	26

Index 29

sharpshootR-package *A collection of functions to support soil survey*

Description

A collection of functions to support soil survey

aggregateColorPlot *Plot aggregate soil color data*

Description

Generate a plot from summaries generated by `aqp::aggregateColor()`.

Usage

```
aggregateColorPlot(x, print.label=TRUE, label.font = 1,
  label.cex = 0.65, buffer.pct = 0.02, print.n.hz=FALSE,
  rect.border='black', horizontal.borders=FALSE, ...)
```

Arguments

<code>x</code>	a list, results from <code>aqp::aggregateColor()</code>
<code>print.label</code>	print Munsell color labels inside of rectangles, when they fit
<code>label.font</code>	font specification for color labels
<code>label.cex</code>	font size for color lables
<code>buffer.pct</code>	extra space between labels and color rectangles

print.n.hz optionally print the number of horizons
 rect.border color for rectangle border
 horizontal.borders
 optionally add horizontal borders between bands of color
 ... additional arguments passed to plot

Details

See examples.

Author(s)

D.E. Beaudette

Examples

```

## Not run:
data(loafercreek, package = 'soilDB')

# generalize horizon names using REGEX rules
n <- c('Oi', 'A', 'BA', 'Bt1', 'Bt2', 'Bt3', 'Cr', 'R')
p <- c('O', '^A$|Ad|Ap|AB', 'BA$|Bw',
      'Bt1$|^B$', '^Bt$|^Bt2$', '^Bt3|^Bt4|CBt$|BCt$|2Bt|2CB$|^C$', 'Cr', 'R')
loafercreek$genhz <- generalize.hz(loafercreek$hname, n, p)

# remove non-matching generalized horizon names
loafercreek$genhz[loafercreek$genhz == 'not-used'] <- NA
loafercreek$genhz <- factor(loafercreek$genhz)

# aggregate color data, this function is from the `aqp` package
a <- aggregateColor(loafercreek, 'genhz')

# plot
par(mar=c(1,4,4,1))
aggregateColorPlot(a, print.n.hz = TRUE)

## End(Not run)

```

amador

SSURGO Data Associated with the Amador Soil Series

Description

SSURGO Data Associated with the Amador Soil Series

Usage

```
data(amador)
```

Format

A subset of data taken from the "component" table of SSURGO

mukey map unit key

compname component name

compct_r component percentage

Source

USDA-NRCS SSURGO Database

aspect.plot	<i>Plot Aspect Data</i>
-------------	-------------------------

Description

Plot a graphical summary of multiple aspect measurements on a circular diagram.

Usage

```
aspect.plot(p, q=c(0.05, 0.5, 0.95), p.bins = 60, p.bw = 30, stack=TRUE,
p.axis = seq(0, 350, by = 10), plot.title = NULL,
line.col='RoyalBlue', line.lwd=1, line.lty=2,
arrow.col=line.col, arrow.lwd=1, arrow.lty=1,
arrow.length=0.15,
...)
```

Arguments

p	a vector of aspect angles in degrees, measured clock-wise from North
q	a vector of desired quantiles
p.bins	number of bins to use for circular histogram
p.bw	bandwidth used for circular density estimation
stack	TRUE/FALSE, should the individual points be stacked into p.bins number of bins and plotted
p.axis	a sequence of integers (degrees) describing the circular axis
plot.title	an informative title
line.col	density line color
line.lwd	density line width
line.lty	density line line style
arrow.col	arrow color
arrow.lwd	arrow line width
arrow.lty	arrow line style
arrow.length	arrow head length
...	further arguments passed to plot.circular

Details

Spread and central tendency are depicted with a combination of circular histogram and kernel density estimate. The circular mean, and relative confidence in that mean are depicted with an arrow: longer arrow lengths correspond to greater confidence in the mean.

Note

Manual adjustment of `p.bw` may be required in order to get an optimal circular density plot. This function requires the package `circular`, version 0.4-7 or later.

Author(s)

D.E. Beaudette

Examples

```
# simulate some data
p.narrow <- runif(n=25, min=215, max=280)
p.wide <- runif(n=25, min=0, max=270)

# set figure margins to 0, 2-column plot
par(mar=c(0,0,0,0), mfcol=c(1,2))

# plot
aspect.plot(p.narrow, p.bw=10, plot.title='Soil A', pch=21, col='black', bg='RoyalBlue')
aspect.plot(p.wide, p.bw=10, plot.title='Soil B', pch=21, col='black', bg='RoyalBlue')
```

CDEC.snow.courses

CDEC Snow Course List

Description

The CDEC snow course list, updated 2014

Usage

```
data(CDEC.snow.courses)
```

Format

A data frame with 261 observations on the following 8 variables.

`course_number` course number

`name` connotative course label

`id` course ID

`elev_feet` course elevation in feet

`latitude` latitude

```

longitude longitude
april.1.Avg.inches average inches of snow as of April 1st
agency responsible agency

```

Source

Data were scraped from <http://cdec.water.ca.gov/misc/SnowCourses.html>, 2014.

Examples

```

data(CDEC.snow.courses)
head(CDEC.snow.courses)

```

CDECquery

Get water-related data (California only) from the CDEC website.

Description

Get water-related data (California only) from the CDEC website.

Usage

```
CDECquery(id, sensor, interval = "D", start, end)
```

Arguments

id	station ID (e.g. 'spw'), see details
sensor	the sensor ID (e.g. 45), see details
interval	character, 'D' for daily, 'H' for hourly, 'M' for monthly
start	starting date, in the format 'YYYY-MM-DD'
end	ending date, in the format 'YYYY-MM-DD'

Details

1. Station IDs can be found here: <http://cdec.water.ca.gov/staInfo.html>
2. Sensor IDs can be found using this URL: http://cdec.water.ca.gov/cgi-progs/queryCSV?station_id=, followed by the station ID.
3. Reservoir capacities can be found here: <http://cdec.water.ca.gov/misc/resinfo.html>
4. A new interactive map of CDEC stations can be found here: <http://cdec.water.ca.gov/cdecstation/>

Value

a data.frame object with the following fields: 'datetime', 'year', 'month', 'value'.

Author(s)

D.E. Beaudette

References<http://cdec.water.ca.gov/queryCSV.html>**See Also**[CDECsnowQuery](#)**Examples**

```
## Not run:
library(latticeExtra)
library(plyr)
library(e1071)

# get daily resevoir storage (ac. ft) from Pinecrest, New Melones and Lyons resevoirs
pinecrest <- CDECquery(id='swb', sensor=15, interval='D', start='2012-09-01', end='2015-01-01')
new.melones <- CDECquery(id='nml', sensor=15, interval='D', start='2012-09-01', end='2015-01-01')
lyons <- CDECquery(id='lys', sensor=15, interval='D', start='2012-09-01', end='2015-01-01')

# compute storage capacity
pinecrest$capacity <- pinecrest$value / 18312 * 100
new.melones$capacity <- new.melones$value / 2400000 * 100
lyons$capacity <- lyons$value / 6228 * 100

# combine
g <- make.groups(new.melones, lyons, pinecrest)

# reasonable date scale
r <- range(g$datetime)
s.r <- seq(from=r[1], to=r[2], by='1 month')

# better colors
tps <- list(superpose.line=list(lwd=2, col=brewer.pal(n=3, name='Set1'))))

# plot
xyplot(capacity ~ datetime, groups=which, data=g, type='l',
       xlab='', ylab='Capacity (
       scales=list(x=list(at=s.r, labels=format(s.r, "
       auto.key=list(columns=3, lines=TRUE, points=FALSE),
       par.settings=tps,
       panel=function(...) {
         panel.abline(h=seq(0, 100, by=10), col='grey')
         panel.abline(v=s.r, col='grey')
         panel.xyplot(...)
       })

##
```

```

# New Melones monthly data, retrieve as far back in time as possible
new.melones.monthly <- CDECquery(id='nml', sensor=15, interval='M',
start='1900-01-01', end='2015-01-01')

# convert to pct. capacity
new.melones.monthly$capacity <- new.melones.monthly$value / 2400000 * 100

# make a nice color ramp function
cols <- colorRampPalette(brewer.pal(9, 'Spectral'),
space='Lab', interpolate='spline')

# plot, each pixel is colored by the total precip by year/month
levelplot(capacity ~ year * month, data=new.melones.monthly, col.regions=cols, xlab='',
ylab='', scales=list(x=list(tick.number=20)), main='New Melones Capacity (

##
# get daily precip totals from Stan Powerhouse
x <- CDECquery(id='spw', sensor=45, interval='D', start='1900-01-01', end='2015-01-01')

# compute total precip by year/month
a <- ddply(x, c('year', 'month'), summarize, s=sum(value, na.rm=TRUE))

# convert monthly precipitation values into Z-scores by month
a.scaled <- ddply(a, 'month', summarize, year=year, scaled.ppt=scale(s))

# make a nice color ramp function, scaled by the skewness of the underlying distribution
cols <- colorRampPalette(brewer.pal(9, 'Spectral'),
space='Lab', interpolate='spline', bias=skewness(a.scaled$scaled.ppt, na.rm=TRUE))

# plot, each pixel is colored by the total precip by year/month
levelplot(scaled.ppt ~ year * month, data=a.scaled, col.regions=cols, xlab='',
ylab='', scales=list(x=list(tick.number=10)),
main='Monthly Total Precipitation (as z-score) SPW')

##
# get pre-aggregated monthly data from Sonora RS
x <- CDECquery(id='sor', sensor=2, interval='M', start='1900-01-01', end='2015-01-01')

# make a nice color ramp function, scaled by the skewness of the underlying distribution
cols <- colorRampPalette(brewer.pal(9, 'Spectral'), space='Lab',
interpolate='spline', bias=skewness(x$value, na.rm=TRUE))

# plot
levelplot(value ~ year * month, data=x, col.regions=cols, xlab='',
ylab='', scales=list(x=list(tick.number=20)), main='Monthly Total Precipitation (inches) SOR')

```



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

CDECsnowQuery *Get snow survey data (California only) from the CDEC website.*

Description

Get snow survey data (California only) from the CDEC website.

Usage

```
CDECsnowQuery(course, start_yr, end_yr)
```

Arguments

course	integer, course number (e.g. 129)
start_yr	integer, the starting year (e.g. 2010)
end_yr	integer, the ending year (e.g. 2013)

Details

This function downloads data from the CDEC website, therefore an internet connection is required. The 'SWE' column contains adjusted SWE if available ('Adjusted' column), otherwise the reported SWE is used ('Water' column).

Value

a data.frame object, see examples

Note

Snow course locations, ID numbers, and other information can be found here: <http://cdec.water.ca.gov/misc/SnowCourses.html>

Author(s)

D.E. Beaudette

References

<http://cdec.water.ca.gov/cgi-progs/snowQuery>

Examples

```
## Not run:  
# get data for course numbe 129  
x <- CDECsnowQuery(course=129, start_yr=2010, end_yr=2011)  
  
## End(Not run)
```

component.adj.matrix *Create an adjacency matrix from a data.frame of component data*

Description

Create an adjacency matrix from SSURGO component data

Usage

```
component.adj.matrix(d, mu='mukey', co='compname', wt='compct_r',
  method='community.matrix', standardization='max', metric='jaccard',
  rm.orphans=TRUE, similarity=TRUE)
```

Arguments

d	a data.frame, typically of SSURGO data
mu	name of the column containing the map unit ID (typically 'mukey')
co	name of the column containing the component ID (typically 'compname')
wt	name of the column containing the component weight percent (typically 'compct_r')
method	one of either: 'community.matrix', or 'occurrence'; see details
standardization	community matrix standardization method, passed to decostand
metric	community matrix dissimilarity metric, passed to vegdist
rm.orphans	logical, should map units with a single component be omitted? (typically yes)
similarity	logical, return a similarity matrix? (if FALSE, a distance matrix is returned)

Details

Pending...

Value

a similarity matrix / adjacency matrix suitable for use with igraph functions or anything else that can accommodate a `_similarity_matrix`.

Author(s)

D.E. Beaudette

Examples

```
# load sample data set
data(amador)

# convert into adjacency matrix
m <- component.adj.matrix(amador)

# plot network diagram, with Amador soil highlighted
plotSoilRelationGraph(m, s='amador')
```

diagnosticPropertyPlot

Diagnostic Property Plot

Description

Generate a graphical description of the presence/absence of soil diagnostic properties.

Usage

```
diagnosticPropertyPlot(f, v, k, grid.label='pedon_id', dend.label='pedon_id')
diagnosticPropertyPlot2(f, v, k, grid.label='pedon_id')
```

Arguments

f	a SoilProfileCollection object
v	a character vector of site-level attribute names that are boolean (e.g. TRUE/FALSE) data
k	an integer, number of groups to highlight
grid.label	the name of a site-level attribute (usually unique) annotating the y-axis of the grid
dend.label	the name of a site-level attribute (usually unique) annotating dendrogram terminal leaves

Details

This function attempts to display several pieces of information within a single figure. First, soil profiles are sorted according to the presence/absence of diagnostic features named in v. Second, these diagnostic features are sorted according to their distribution among soil profiles. Third, a binary grid is established with row-ordering of profiles based on step 1 and column-ordering based on step 2. Blue cells represent the presence of a diagnostic feature. Soils with similar diagnostic features should 'clump' together. See examples below.

Value

a list is silently returned by this function, containing:

`rd` a data.frame containing IDs and grouping code

`profile.order` a vector containing the order of soil profiles (row-order in figure), according to diagnostic property values

`var.order` a vector containing the order of variables (column-order in figure), according to their distribution among profiles

Author(s)

D.E. Beaudette and J.M. Skovlin

See Also

[multinomial2logical](#)

Examples

```
## Not run:
# sample data, an SPC
data(gopheridge, package='soilDB')

# get depth class
sdc <- getSoilDepthClass(gopheridge)
site(gopheridge) <- sdc

# diagnostic properties to consider, no need to convert to factors
v <- c('lithic.contact', 'paralithic.contact', 'argillic.horizon',
       'cambic.horizon', 'ochric.epipedon', 'mollic.epipedon', 'very.shallow',
       'shallow', 'mod.deep', 'deep', 'very.deep')

# base graphics
x <- diagnosticPropertyPlot(gopheridge, v, k=5)

# lattice graphics
x <- diagnosticPropertyPlot2(gopheridge, v, k=3)

# check output
str(x)

## End(Not run)
```

dist.along.grad	<i>Compute Euclidean distance along a gradient.</i>
-----------------	---

Description

This function computes Euclidean distance along points aligned to a given gradient (e.g. elevation).

Usage

```
dist.along.grad(coords, var, grad.scaled.min, grad.scaled.max)
```

Arguments

coords	a matrix of x and y coordinates in some projected coordinate system
var	a vector of the same length as coords, describing the gradient of interest
grad.scaled.min	min value of rescaled gradient values
grad.scaled.max	max value of rescaled gradient values

Details

This function is primarily intended for use within [plotTransect](#).

Value

A data.frame object:

scaled.grad scaled gradient values

scaled.distance cumulative distance, scaled to the interval of 0.5, $nrow(coords) + 0.5$

distance cumulative distance computed along gradient, e.g. transect distance

variable sorted gradient values

x x coordinates, ordered by gradient values

y y coordinate, ordered by gradient values

grad.order a vector index describing the sort order defined by gradient values

Note

This function is very much a work in progress, ideas welcome.

Author(s)

D.E. Beaudette

See Also

[plotTransect](#)

dueling.dendrograms *Dueling Dendrograms*

Description

Graphically compare two related dendrograms

Usage

```
dueling.dendrograms(p.1, p.2, lab.1 = "D1",  
lab.2 = "D2", cex.nodelabels=0.75, arrow.length=0.05)
```

Arguments

p.1	left-hand phylo-class dendrogram
p.2	right-hand phylo-class dendrogram
lab.1	left-hand title
lab.2	right-hand title
cex.nodelabels	character expansion size for node labels
arrow.length	arrow head size

Details

Connector arrows are used to link nodes from the left-hand dendrogram to the right-hand dendrogram.

Author(s)

D. E. Beaudette

Examples

```
library(aqp)  
library(cluster)  
library(ape)  
  
# load sample dataset from aqp package  
data(sp3)  
  
# promote to SoilProfileCollection  
depths(sp3) <- id ~ top + bottom  
  
# compute dissimilarity using different sets of variables  
# note that these are rescaled to the interval [0,1]  
d.1 <- profile_compare(sp3, vars=c('clay', 'cec'), k=0, max_d=100, rescale.result=TRUE)  
d.2 <- profile_compare(sp3, vars=c('clay', 'L'), k=0, max_d=100, rescale.result=TRUE)
```

```

# cluster via divisive hierarchical algorithm
# convert to 'phylo' class
p.1 <- as.phylo(as.hclust(diana(d.1)))
p.2 <- as.phylo(as.hclust(diana(d.2)))

# graphically compare two dendrograms
dueling.dendrograms(p.1, p.2, lab.1='clay and CEC', lab.2='clay and L')

# graphically check the results of ladderize() from ape package
dueling.dendrograms(p.1, ladderize(p.1), lab.1='standard', lab.2='ladderized')

# sanity-check: compare something to itself
dueling.dendrograms(p.1, p.1, lab.1='same', lab.2='same')

# graphically compare diana() to agnes() using d.2
dueling.dendrograms(as.phylo(as.hclust(diana(d.2))),
as.phylo(as.hclust(agnes(d.2))), lab.1='diana', lab.2='agnes')

```

generateLineHash	<i>Generate a unique ID for line segments</i>
------------------	---

Description

Generate a unique ID for a line segment, based on the non-cryptographic murmur32 hash.

Usage

```
generateLineHash(x, precision=-1, algo='murmur32')
```

Arguments

x	a SpatialLinesDataFrame object, with 1 line segment per feature (e.g. simple features)
precision	digits are rounded to this many places to the right (negative) or left (positive) of the decimal place
algo	hash function algorithm

Details

The input SpatialLinesDataFrame object must NOT contain multi-part features. The precision specified should be tailored to the coordinate system in use and the snapping tolerance used to create join decision line segments. A precision of 4 is reasonable for geographic coordinates (snapping tolerance of 0.0001 degrees or ~ 10 meters). A precision of -1 (snapping tolerance of 10 meters) is reasonable for projected coordinate systems with units in meters.

Value

A vector of unique IDs created from the hash of line segment start and end vertex coordinates. Unique IDs are returned in the order of records of x and can therefore be saved into a new column of the associated attribute table.

Note

An error is issued if any non-unique IDs are generated. This could be caused by using coordinates that do not contain enough precision for unique hashing.

Author(s)

D.E. Beaudette

hillslope.probability *Hillslope Probability via SDA*

Description

Hillslope position probability estimates from the SDA query service (SSURGO)

Usage

```
hillslope.probability(s)
```

Arguments

s a character vector of soil series names, in lower-case

Details

This function constructs and executes a query that is sent to the <http://sdmdataaccess.nrcs.usda.gov> webservice. Further information on the SDA webservice and query examples can be found at <http://sdmdataaccess.nrcs.usda.gov/QueryHelp.aspx>

Value

A data.frame object with rows representing soil series, and columns representing probability estimates of that series occurring along the standard 2D hillslope positions.

Note

Probability values are computed from SSURGO data.

Author(s)

D.E. Beaudette

Examples

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

# soil series of interest
s <- c('auburn', 'pardee', 'amador', 'pentz')

# generate hillslope probability table
hillslope.probability(s)

## End(Not run)
```

joinAdjacency	<i>Join Document Adjacency</i>
---------------	--------------------------------

Description

Convert a set of line segment "join decisions" into a weighted adjacency matrix describing which map unit symbols touch.

Usage

```
joinAdjacency(x, vars = c("l_musym", "r_musym"))
```

Arguments

x	a SpatialLinesDataFrame object, with 1 line segment per feature (e.g. simple features)
vars	a vector of two characters naming columns containing "left", and "right" map unit symbols

Value

A weighted adjacency matrix is returned, suitable for plotting directly with `plotSoilRelationGraph`.

Author(s)

D.E. Beaudette

See Also

[plotSoilRelationGraph](#)

multinomial2logical *Convert multinomial to logical matrix*

Description

Convert a single multinomial, site-level attribute from a SoilProfileCollection into a matrix of corresponding logical values. The result contains IDs from the SoilProfileCollection and can easily be joined to the original site-level data.

Usage

```
multinomial2logical(x, v)
```

Arguments

x a SoilProfileCollection object
v the name of a site-level attribute that is a factor with more than 2 levels

Value

A data.frame with IDs in the first column, and as many columns of logical vectors as there were levels in v. See examples.

Author(s)

D.E. Beaudette

See Also

[diagnosticPropertyPlot](#)

Examples

```
## Not run:  
# sample data, an SPC  
data(loafercreek, package='soilDB')  
  
# convert to factor  
loafercreek$hillslope_pos <- factor(loafercreek$hillslope_pos,  
levels=c('Toeslope', 'Footslope', 'Backslope', 'Shoulder', 'Summit'))  
  
# convert to logical matrix  
hp <- multinomial2logical(loafercreek, 'hillslope_pos')  
  
# join-in to site data  
site(loafercreek) <- hp  
  
# variable names  
v <- c('lithic.contact', 'paralithic.contact',
```

```
'argillic.horizon', 'Toeslope', 'Footslope',
'Backslope', 'Shoulder', 'Summit')

# visualize with some other diagnostic features
x <- diagnosticPropertyPlot(loafercreek, v, k=5,
grid.label='bedrock_kind', dend.label='pedon_id')

## End(Not run)
```

plotAvailWater

Visual Demonstration of Available Soil Water

Description

Generate a simplistic diagram of the various fractions of water held within soil pore-space.

Usage

```
plotAvailWater(x, width = 0.25, cols = c(grey(0.5),
"DarkGreen", "LightBlue", "RoyalBlue"), name.cex = 0.8)
```

Arguments

x	a data.frame containing sample names and water retention data, see examples below
width	vertical width of each bar graph
cols	a vector of colors used to symbolize 'solid phase', 'unavailable water', 'available water', and 'gravitational water'
name.cex	character scaling of horizon names, printed on left-hand side of figure

Author(s)

D.E. Beaudette

Examples

```
# demonstration
s <- data.frame(
name=c('loamy sand', 'sandy loam', 'silt loam', 'clay loam'),
pwp=c(0.05, 0.1, 0.18, 0.2),
fc=c(0.1, 0.2, 0.38, 0.35),
sat=c(0.25, 0.3, 0.45, 0.4))
s$solid <- with(s, 1-sat)

par(mar=c(5, 6, 0.5, 0.5))
plotAvailWater(s, name.cex=1.25)

# use some real data from SSURGO
```

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

q <- "SELECT hzdept_r as hztop, hzdepb_r as hzbottom,
hzname as name, wsatiated_r/100.0 as sat,
wthirdbar_r/100.0 as fc, wfifteenbar_r/100.0 as pwp, awc_r as awc
FROM chorizon
WHERE cokey IN (SELECT cokey from component where compname = 'dunstone')
AND wsatiated_r IS NOT NULL
ORDER BY cokey, hzdept_r ASC;"

x <- SDA_query(q)
x <- unique(x)
x <- x[order(x$name), ]
x$solid <- with(x, 1-sat)

par(mar=c(5, 5, 0.5, 0.5))
plotAvailWater(x)

## End(Not run)
```

plotSoilRelationGraph *Plot a component relation graph*

Description

Plot a component relation graph based on an adjacency or similarity matrix.

Usage

```
plotSoilRelationGraph(m, s='', plot.style='network', spanning.tree=NULL,
del.edges=NULL, vertex.scaling.factor=2, edge.scaling.factor=1,
edge.transparency=1, edge.col=grey(0.5), edge.highlight.col='royalblue',
g.layout=layout.fruchterman.reingold,
...)
```

Arguments

m	adjacency matrix
s	central component; an empty character string is interpreted as no central component
plot.style	plot style ('network', or 'dendrogram'), or 'none' for no graphical output
spanning.tree	plot the minimum or maximum spanning tree ('min', 'max'), or, max spanning tree plus edges with weight greater than the n-th quantile specified in 'spanning.tree'. See details and examples.
del.edges	optionally delete edges with weights less than the specified quantile (0-1)

```

vertex.scaling.factor      scaling factor applied to vertex size
edge.scaling.factor       optional scaling factor applied to edge width
edge.transparency         optional transparency setting for edges (0-1)
edge.col                  edge color, applied to all edges
edge.highlight.col       edge color applied to all edges connecting to component named in s
g.layout                  an igraph layout function, defaults to layout.fruchterman.reingold
...                       further arguments passed to plotting function

```

Details

Vertex size is based on a normalized index of connectivity: $size = \sqrt{\text{degree}(g)/\text{max}(\text{degree}(g))} * \text{scaling.factor}$. Edge width can be optionally scaled by edge weight by specifying an `edge.scaling.factor` value. The maximum spanning tree represents a sub-graph where the sum of edge weights are maximized. The minimum spanning tree represents a sub-graph where the sum of edge weights are minimized. The maximum spanning tree is likely a more useful simplification of the full graph, in which only the strongest relationships (e.g. most common co-occurrences) are preserved.

The maximum spanning tree + edges with weights > n-th quantile is an experimental hybrid. The 'backbone' of the graph is created by the maximum spanning tree, and augmented by 'strong' auxillary edges— defined by a value between 0 and 1.

Value

an igraph 'graph' object is invisibly returned

Note

This function is a work in progress, ideas welcome.

Author(s)

D.E. Beaudette

Examples

```

# load sample data set
data(amador)

# create weighted adjacency matrix (see ?component.adj.matrix for details)
m <- component.adj.matrix(amador)

# plot network diagram, with Amador soil highlighted
plotSoilRelationGraph(m, s='amador')

# dendrogram representation
plotSoilRelationGraph(m, s='amador', plot.style='dendrogram')

```

```

# compare methods
m.o <- component.adj.matrix(amador, method='occurrence')

par(mfcol=c(1,2))
plotSoilRelationGraph(m, s='amador', plot.style='dendrogram')
title('community matrix')
plotSoilRelationGraph(m.o, s='amador', plot.style='dendrogram')
title('occurrence')

# investigate max spanning tree
plotSoilRelationGraph(m, spanning.tree='max')

# investigate max spanning tree + edges with weights > 75-th pctlile
plotSoilRelationGraph(m, spanning.tree=0.75)

## Not run:
# get similar data from soilweb, for the Pardee series
s <- 'pardee'
u <- url(URLencode(paste(
'http://casoilresource.lawr.ucdavis.edu/soil_web/reflector_api/soils.php?',
'what=soil_series_component_query&q_string=', s, sep='')))

# fetch data
d <- read.table(u, sep='|', header=TRUE, stringsAsFactors=FALSE)

# normalize component names
d$compname <- tolower(d$compname)

# keep only major components
d <- subset(d, subset=compkind == 'Series')

# build adj. matrix and plot
m <- component.adj.matrix(d)
plotSoilRelationGraph(m, s=s, plot.style='dendrogram')

# alter plotting style, see ?plot.phylo
plotSoilRelationGraph(m, s=s, plot.style='dendrogram', type='fan')
plotSoilRelationGraph(m, s=s, plot.style='dendrogram', type='unrooted', use.edge.length=FALSE)

## End(Not run)

```

plotTransect

Plot a collection of Soil Profiles linked to their position along some gradient (e.g. transect).

Description

Plot a collection of Soil Profiles linked to their position along some gradient (e.g. transect).

Usage

```
plotTransect(s, grad.var.name, transect.col = "RoyalBlue",
  tick.number=7, y.offset = 100,
  scaling.factor = 0.5,
  distance.axis.title = "Distance Along Transect (km)",
  crs = NULL, grad.axis.title = NULL, ...)
```

Arguments

<code>s</code>	a <code>SoilProfileCollection</code> object
<code>grad.var.name</code>	the name of a site-level attribute containing gradient values
<code>transect.col</code>	color used to plot gradient (transect) values
<code>tick.number</code>	number of desired ticks and labels on the gradient axis
<code>y.offset</code>	vertical offset used to position profile sketches
<code>scaling.factor</code>	scaling factor used to adjust profile sketches
<code>distance.axis.title</code>	a title for the along-transect distances
<code>crs</code>	an optional CRS object used to convert coordinates into a planar system
<code>grad.axis.title</code>	a title for the gradient axis
<code>...</code>	further arguments passed to <code>plotSPC</code>

Details

Depending on the nature of your `SoilProfileCollection` and associated gradient values, it may be necessary to tinker with figure margins, `y.offset` and `scaling.factor`.

Value

An invisibly-returned `data.frame` object:

- scaled.grad** scaled gradient values
- scaled.distance** cumulative distance, scaled to the interval of 0.5, $nrow(coords) + 0.5$
- distance** cumulative distance computed along gradient, e.g. transect distance
- variable** sorted gradient values
- x** x coordinates, ordered by gradient values
- y** y coordinate, ordered by gradient values
- grad.order** a vector index describing the sort order defined by gradient values

Note

This function is very much a work in progress, ideas welcome!

Author(s)

D.E. Beaudette

Examples

```
## Not run:
data(loafercreek, package='soilDB')

x <- loafercreek[1:10, ]
coordinates(x) <- ~ x_std + y_std
proj4string(x) <- '+proj=longlat +datum=NAD83'

par(mar=c(4,3,1,1))
plotTransect(x, 'elev_field', crs=CRS('+proj=utm +zone=10 +datum=NAD83'), max.depth=185)

## End(Not run)
```

polygonAdjacency

Evaluate Spatial Adjacency of SpatialPolygonsDataFrame Objects

Description

This function utilizes the ‘spdep’ and ‘igraph’ packages to evaluate several measures of spatial connectivity.

Usage

```
polygonAdjacency(x, v='MUSYM', ...)
```

Arguments

x	a SpatialPolygonsDataFrame object
v	name of the field in the attribute table to use when searching for ‘common lines’, see details
...	additional arguments passed to spdep::poly2nb

Details

Pending.

Value

A list object containing:

commonLines An integer vector of feature IDs, that share a common boundary and attribute v.commonLines. Sometimes referred to as "common soil lines".

adjMat A weighted adjacency matrix

Author(s)

D.E. Beaudette

Examples

```

## Not run:
library(spdep)
library(rgdal)
library(igraph)
library(sharpshootR)

# load some data
x <- readOGR(dsn='L:/CA630/FG_CA630_OFFICIAL.gdb', layer='ca630_a', stringsAsFactors=FALSE)

# remove NOTCOM, DA, and empty (non-NA) symbols
x <- x[which(! x$MUSYM

# compute spatial adjacency summary
res <- polygonAdjacency(x, v='MUSYM')

# graphical check: slow for large number of features
plot(x)
plot(x[res$commonLines, ], col='red', add=TRUE)

# save to SHP
writeOGR(x[res$commonLines, ], dsn='.',
layer='common-soil-lines', driver='ESRI Shapefile',
overwrite_layer=TRUE)

# plot spatial adjacency information
par(mar=c(0,0,0,0))
plotSoilRelationGraph(res$adjMat,
vertex.scaling.factor = 1)
plotSoilRelationGraph(res$adjMat, spanning.tree='max',
edge.scaling.factor=0.1, vertex.scaling.factor=1)

## End(Not run)

```

sample.by.poly

Sample a Polygon at Fixed Density

Description

Generate sampling points within a SpatialPolygon object, according to a specified sampling density.

Usage

```

sample.by.poly(p, n.pts.per.ac=1, min.samples=5,
sampling.type='hexagonal', iterations=10, p4s=NULL)

```

Arguments

<code>p</code>	a Polygon object, with coordinates in a projected CRS
<code>n.pts.per.ac</code>	requested sampling density in points per acre (results will be close)
<code>min.samples</code>	minimum requested number of samples per polygon
<code>sampling.type</code>	sampling type, see <code>spsample</code>
<code>iterations</code>	number of tries that <code>spsample</code> will attempt
<code>p4s</code>	a qualified proj4string that will be assigned to sampling points

Details

This function is typically accessed via some kind of helper function such as [constantDensitySampling](#).

Value

A `SpatialPoints` object.

Note

This function expects that the Polygon object has coordinates associated with a projected CRS—e.g. units of meters.

Author(s)

D.E. Beaudette

See Also

[constantDensitySampling](#)

SoilTaxonomyDendrogram

Soil Taxonomy Dendrogram

Description

Plot a dendrogram based on the first 4 levels of Soil Taxonomy, with soil profiles hanging below. A dissimilarity matrix is computed using Gower's distance metric for nominal-scale variables, based on order, sub order, great group, and subgroup level taxa. See the Details and Examples sections below for more information.

Usage

```
SoilTaxonomyDendrogram(spc, name = "hzname", max.depth = 150,  
n.depth.ticks = 6, scaling.factor = 0.015, cex.names = 0.75,  
cex.id = 0.75, axis.line.offset = -4, width = 0.1, y.offset = 0.5,  
cex.taxon.labels = 0.66)
```

Arguments

<code>spc</code>	a SoilProfileCollection object, see details
<code>name</code>	column name containing horizon names
<code>max.depth</code>	depth at which profiles are truncated for plotting
<code>n.depth.ticks</code>	suggested number of ticks on the depth axis
<code>scaling.factor</code>	scaling factor used to convert depth units into plotting units
<code>cex.names</code>	character scaling for horizon names
<code>cex.id</code>	character scaling for profile IDs
<code>axis.line.offset</code>	horizontal offset for depth axis
<code>width</code>	width of profiles
<code>y.offset</code>	vertical offset between dendrogram and profiles
<code>cex.taxon.labels</code>	character scaling for taxonomic information

Details

This function looks for specific site-level attributes named: `soilorder`, `suborder`, `greatgroup`, and `subgroup`.

Value

An invisibly-returned list containing:

`dist` pair-wise dissimilarity matrix

Author(s)

D.E. Beaudette

References

D.E. Beaudette, P. Roudier and A.T. O'Geen. 2012. Algorithms for Quantitative Pedology, a Toolkit for Soil Scientists. *Computers & Geosciences*: 52: 258–268. (doi: 10.1016/j.cageo.2012.10.020)

<http://aqp.r-forge.r-project.org/>

Examples

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

# 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
h <- fetchOSD(s.list)
```

```
# plot dendrogram + profiles
SoilTaxonomyDendrogram(h)

# again, this time save the pair-wise dissimilarity matrix
# note that there isn't a lot of discrimination between soils
(d <- SoilTaxonomyDendrogram(h))

## End(Not run)
```

Index

*Topic **datasets**

amador, 3
CDEC.snow.courses, 5

*Topic **hplots**

aggregateColorPlot, 2
aspect.plot, 4
diagnosticPropertyPlot, 11
dueling.dendrograms, 14
plotAvailWater, 19
plotTransect, 22
SoilTaxonomyDendrogram, 26

*Topic **hplot**

plotSoilRelationGraph, 20

*Topic **manip**

component.adj.matrix, 10
dist.along.grad, 13
generatelineHash, 15
hillslope.probability, 16
joinAdjacency, 17
multinomial2logical, 18
polygonAdjacency, 24
sample.by.poly, 25

aggregateColorPlot, 2

amador, 3

aspect.plot, 4

CDEC.snow.courses, 5

CDECquery, 6

CDECsnowQuery, 7, 9

component.adj.matrix, 10

constantDensitySampling, 26

constantDensitySampling
(sample.by.poly), 25

diagnosticPropertyPlot, 11, 18

diagnosticPropertyPlot2
(diagnosticPropertyPlot), 11

dist.along.grad, 13

dueling.dendrograms, 14

generatelineHash, 15

hillslope.probability, 16

joinAdjacency, 17

multinomial2logical, 12, 18

plotAvailWater, 19

plotSoilRelationChordGraph
(plotSoilRelationGraph), 20

plotSoilRelationGraph, 17, 20

plotTransect, 13, 22

polygonAdjacency, 24

sample.by.poly, 25

sharpshootR (sharpshootR-package), 2

sharpshootR-package, 2

SoilTaxonomyDendrogram, 26