

# Package ‘sharpshootR’

March 22, 2023

**Type** Package

**Encoding** UTF-8

**Title** A Soil Survey Toolkit

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

**Version** 2.1

**Date** 2023-03-21

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

**LazyLoad** yes

**LazyData** true

**License** GPL (>= 3)

**Repository** CRAN

**URL** <https://github.com/ncss-tech/sharpshootR>

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

**Suggests** MASS, spdep, circlize, rvest, xml2, terra, raster, exactextractr, htr, jsonlite, igraph, dendextend, testthat, hydromad (>= 0.9.27), latticeExtra, farver, venn, gower, daymetr, elevatr, Evapotranspiration, zoo, SoilTaxonomy, sf

**Depends** R (>= 3.5.0)

**Imports** grDevices, graphics, methods, stats, utils, aqp, ape, soilDB, cluster, lattice, vegan, reshape2, Hmisc, scales, circular, RColorBrewer, plyr, digest, e1071, stringi, parallel, curl, grid

**Additional\_repositories** <https://hydromad.github.io>

**RoxygenNote** 7.2.3

**NeedsCompilation** no

**Author** Dylan Beaudette [cre, aut],  
Jay Skovlin [aut],  
Stephen Roecker [aut],  
Andrew Brown [aut],  
USDA-NRCS Soil Survey Staff [ctb]

Date/Publication 2023-03-22 16:50:05 UTC

## R topics documented:

sharpshootR-package	3
aggregateColorPlot	4
amador	6
aspect.plot	6
CDEC.snow.courses	8
CDECquery	9
CDECsnowQuery	10
CDEC_StationInfo	11
colorMixtureVenn	12
component.adj.matrix	13
constantDensitySampling	14
dailyWB	15
dailyWB_SSURGO	16
diagnosticPropertyPlot	17
diagnosticPropertyPlot2	19
dist.along.grad	21
dueling.dendrograms	22
ESS_by_Moran_I	23
estimateSoilMoistureState	23
FFD	25
FFDplot	26
formatPLSS	27
generateLineHash	28
geomorphBySoilSeries-SSURGO	29
HenryTimeLine	30
HHM	31
huePositionPlot	32
hydOrder	33
isMineralSoilMaterial	34
iterateHydOrder	35
joinAdjacency	36
LL2PLSS	37
moistureStateProportions	38
moistureStateStats	39
moistureStateThreshold	39
monthlyWB	40
Moran_I_ByRaster	42
multinomial2logical	43
OSDexamples	44
PCP_plot	44
percentileDemo	45
plotAvailWater	46
plotGeomorphCrossSection	49
plotProfileDendrogram	49

plotSoilRelationChordGraph . . . . .	51
plotSoilRelationGraph . . . . .	52
plotTransect . . . . .	55
plotWB . . . . .	58
plotWB_lines . . . . .	60
PLSS2LL . . . . .	62
polygonAdjacency . . . . .	62
prepareDailyClimateData . . . . .	63
prepare_SSURGO_hydro_data . . . . .	64
reconcileOSDGeomorph . . . . .	64
sample.by.poly . . . . .	65
sampleRasterStackByMU . . . . .	66
samplingStability . . . . .	67
simpleWB . . . . .	68
site_photos_kml . . . . .	69
SoilTaxonomyDendrogram . . . . .	70
table5.2 . . . . .	73
vizAnnualClimate . . . . .	74
vizFlatsPosition . . . . .	75
vizGeomorphicComponent . . . . .	76
vizHillslopePosition . . . . .	77
vizMountainPosition . . . . .	78
vizSurfaceShape . . . . .	79
vizTerracePosition . . . . .	80
<b>Index</b>	<b>81</b>

---

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

---

## Description

This package contains mish-mash of functionality and sample data related to the daily business of soil survey operations with the USDA-NRCS. Many of the functions are highly specialized and inherit default arguments from the names used by the various NCSS (National Cooperative Soil Survey) databases. A detailed description of this package with links to associated tutorials can be found at the [project website](#).

---

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,
  label.orientation = c("v", "h"),
  buffer.pct = 0.02,
  print.n.hz = FALSE,
  rect.border = "black",
  horizontal.borders = FALSE,
  horizontal.border.lwd = 2,
  x.axis = TRUE,
  y.axis = TRUE,
  ...
)
```

### Arguments

<code>x</code>	a list, results from <code>aqp::aggregateColor()</code>
<code>print.label</code>	logical, print Munsell color labels inside of rectangles, only if they fit
<code>label.font</code>	font specification for color labels
<code>label.cex</code>	font size for color labels
<code>label.orientation</code>	label orientation, v for vertical or h for horizontal
<code>buffer.pct</code>	extra space between labels and color rectangles
<code>print.n.hz</code>	optionally print the number of horizons below Munsell color labels
<code>rect.border</code>	color for rectangle border
<code>horizontal.borders</code>	optionally add horizontal borders between bands of color
<code>horizontal.border.lwd</code>	line width for horizontal borders
<code>x.axis</code>	logical, add a scale and label to x-axis?
<code>y.axis</code>	logical, add group labels to y-axis?
<code>...</code>	additional arguments passed to plot

**Details**

Tutorial at <http://ncss-tech.github.io/AQP/sharpsshootR/aggregate-soil-color.html>.

**Value**

nothing, function called for graphical output

**Author(s)**

D.E. Beaudette

**Examples**

```
if(require(aqp) &
  require(soilDB)) {

  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$genhz <- generalize.hz(loafercreek$hzone, 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
  op <- par(no.readonly = TRUE)

  par(mar=c(4,4,1,1))

  # vertical labels, the default
  aggregateColorPlot(a, print.n.hz = TRUE)

  # horizontal labels
  aggregateColorPlot(a, print.n.hz = TRUE, label.orientation = 'h')

  par(op)
}
```

---

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

*Plot Aspect Data*

---

**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

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

### 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.

### Value

invisibly returns circular stats

### 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
op <- par(no.readonly = TRUE)
par(mar = c(0,0,0,0), mfcol = c(1,2))

# plot, save circular stats
x <- aspect.plot(p.narrow, p.bw=10, plot.title='Soil A', pch=21, col='black', bg='RoyalBlue')
y <- aspect.plot(p.wide, p.bw=10, plot.title='Soil B', pch=21, col='black', bg='RoyalBlue')

# reset output device options
par(op)

x
```

---

CDEC.snow.courses

*CDEC Snow Course List*

---

**Description**

The CDEC snow course list, updated September 2019

**Usage**

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

**Format**

A data frame with 259 observations on the following 9 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

watershed watershed label



**Source**

Data were scraped from <http://cdec.water.ca.gov/misc/SnowCourses.html>, 2019.

**Examples**

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

---

CDECquery

*Easy Access to the CDEC API*


---

**Description**

A (relatively) simple interface to the CDEC website.

**Usage**

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

**Arguments**

id	station ID (e.g. 'spw'), single value or vector of station IDs, see details
sensor	the sensor ID, single value or vector of sensor numbers, see details
interval	character, 'D' for daily, 'H' for hourly, 'M' for monthly, 'E' for event: see Details.
start	starting date, in the format 'YYYY-MM-DD'
end	ending date, in the format 'YYYY-MM-DD'

**Details**

Sensors that report data on an interval other than monthly ('M'), daily ('D'), or hourly ('H') can be queried with an event interval ('E'). Soil moisture and temperature sensors are an example of this type of reporting. See examples below.

1. Station IDs can be found here: <http://cdec.water.ca.gov/staInfo.html>
- 2a. Sensor IDs can be found using this URL: [http://cdec.water.ca.gov/dynamicapp/staMeta?station\\_id=](http://cdec.water.ca.gov/dynamicapp/staMeta?station_id=), followed by the station ID.
- 2b. Sensor details can be accessed using [CDEC\\_StationInfo](#) with 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>

**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](#) [CDEC\\_StationInfo](#)

---

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). See the [tutorial](#) for examples.

**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>

---

CDEC_StationInfo	<i>CDEC Sensor Details (by Station)</i>
------------------	---

---

**Description**

Query CDEC Website for Sensor Details

**Usage**

CDEC\_StationInfo(s)

**Arguments**

s                      character, a single CDEC station ID (e.g. 'HHM')

**Details**

This function requires the rvest package.

**Value**

A list object containing site metadata, sensor metadata, and possibly comments about the site.

**Author(s)**

D.E. Beaudette

**See Also**

[CDECquery]

---

`colorMixtureVenn`*Create a Venn Diagram of Simulated Color Mixtures*

---

## Description

Create a Venn Diagram of Simulated Color Mixtures

## Usage

```
colorMixtureVenn(  
  chips,  
  w = rep(1, times = length(chips))/length(chips),  
  mixingMethod = "exact",  
  ellipse = FALSE,  
  labels = TRUE,  
  names = FALSE,  
  sncs = 0.85  
)
```

## Arguments

<code>chips</code>	character vector of standard Munsell color notation (e.g. "10YR 3/4")
<code>w</code>	vector of proportions, can sum to any number, must be same length as <code>chips</code>
<code>mixingMethod</code>	approach used to simulate a mixture: see <a href="#">aqp::mixMunsell()</a> for details
<code>ellipse</code>	logical, use alternative ellipse-style (4 or 5 colors only)
<code>labels</code>	logical, print mixture labels
<code>names</code>	logical, print names outside of the "sets"
<code>sncs</code>	scaling factor for set names

## Value

nothing returned, function is called to create graphical output

## Examples

```
if(requireNamespace("venn") & requireNamespace("gower")) {  
  
  chips <- c('10YR 8/1', '2.5YR 3/6', '10YR 2/2')  
  names(chips) <- c("tan", "dark red", "dark brown")  
  
  colorMixtureVenn(chips)  
  colorMixtureVenn(chips, names = TRUE)  
  
  colorMixtureVenn(chips, w = c(1, 1, 1), names = TRUE)  
  colorMixtureVenn(chips, w = c(10, 5, 1), names = TRUE)
```

```
}

```

---

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 = c("community.matrix", "occurrence"),
  standardization = "max",
  metric = "jaccard",
  rm.orphans = TRUE,
  similarity = TRUE,
  return.comm.matrix = FALSE
)
```

### Arguments

d	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 vegan::decostand
metric	community matrix dissimilarity metric, passed to vegan::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)
return.comm.matrix	logical, return pseudo-community matrix? (if TRUE no adjacency matrix is created)

**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**

```
if (requireNamespace("igraph")) {
  # 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')
}
```

---

constantDensitySampling

*Constant Density Sampling*

---

**Description**

Perform sampling at a constant density over all polygons within a SpatialPolygonsDataFrame object.

**Usage**

```
constantDensitySampling(x, polygon.id='pID', parallel=FALSE, cores=NULL,
  n.pts.per.ac=1, min.samples=5, sampling.type='regular')
```

**Arguments**

x	a SpatialPolygonsDataFrame object in a projected CRS with units of meters
polygon.id	name of attribute in x that contains a unique ID for each polygon
parallel	invoke parallel back-end
cores	number of CPU cores to use for parallel operation
n.pts.per.ac	requested sampling density in points per acre (results will be close)
min.samples	minimum requested number of samples per polygon
sampling.type	sampling type

**Value**

a `SpatialPointsDataFrame` object

**Note**

This function expects that `x` has coordinates associated with a projected CRS and units of meters.

**Author(s)**

D.E. Beaudette

**See Also**

[sample.by.poly](#)

---

dailyWB

*Simple Daily Water Balance*

---

**Description**

Simple interface to the hydromad "leaky bucket" soil moisture model, with accommodation for typical inputs from common soil data and climate sources. Critical points along the water retention curve are specified using volumetric water content (VWC): satiation (saturation), field capacity (typically 1/3 bar suction), and permanent wilting point (typically 15 bar suction).

**Usage**

```
dailyWB(x, daily.data, id, MS.style = "default", S_0 = 0.5, M = 0, etmult = 1)
```

**Arguments**

<code>x</code>	<p>data.frame, required columns include:</p> <ul style="list-style-type: none"> <li>• <code>sat</code>: VWC at satiation</li> <li>• <code>fc</code>: VWC at field capacity</li> <li>• <code>pwp</code>: VWC at permanent wilting point</li> <li>• <code>thickness</code>: soil material thickness in cm</li> <li>• <code>a.ss</code>: recession coefficients for subsurface flow from saturated zone, should be &gt; 0 (range: 0-1)</li> <li>• "id"</li> </ul>
<code>daily.data</code>	<p>data.frame, required columns include:</p> <ul style="list-style-type: none"> <li>• <code>date</code>: Date class representation of dates</li> <li>• <code>PPT</code>: daily total, precipitation in mm</li> <li>• <code>PET</code>: daily total, potential ET in mm</li> </ul>
<code>id</code>	character, name of column in <code>x</code> that is used to identify records
<code>MS.style</code>	moisture state classification style, see <a href="#">estimateSoilMoistureState</a>

S_0	fraction of water storage filled at time = 0 (range: 0-1)
M	fraction of area covered by deep-rooted vegetation
etmult	multiplier for PET

**Value**

a `data.frame`

**References**

Farmer, D., M. Sivapalan, Farmer, D. (2003). Climate, soil and vegetation controls upon the variability of water balance in temperate and semiarid landscapes: downward approach to water balance analysis. *Water Resources Research* 39(2), p 1035.

Bai, Y., T. Wagener, P. Reed (2009). A top-down framework for watershed model evaluation and selection under uncertainty. *Environmental Modelling and Software* 24(8), pp. 901-916.

---

dailyWB_SSURGO	<i>Perform daily water balance modeling using SSURGO and DAYMET</i>
----------------	---

---

**Description**

Pending.

**Usage**

```
dailyWB_SSURGO(
  x,
  cokeys = NULL,
  start = 1988,
  end = 2018,
  modelDepth = 100,
  MS.style = "default",
  a.ss = 0.1,
  S_0 = 0.5,
  bufferRadiusMeters = 1
)
```

**Arguments**

x	sf object representing a single point
cokeys	vector of component keys to use
start	starting year (limited to DAYMET holdings)
end	ending year (limited to DAYMET holdings)
modelDepth	soil depth used for water balance, see details
MS.style	moisture state classification style, see <a href="#">estimateSoilMoistureState</a>



a.ss	recession coefficients for subsurface flow from saturated zone, should be > 0 (range: 0-1)
S_0	fraction of water storage filled at time = 0 (range: 0-1)
bufferRadiusMeters	spatial buffer (meters) applied to x for the look-up of SSURGO data

**Value**

data.frame of daily water balance results

**Author(s)**

D.E. Beaudette

**References**

Farmer, D., M. Sivapalan, Farmer, D. (2003). Climate, soil and vegetation controls upon the variability of water balance in temperate and semiarid landscapes: downward approach to water balance analysis. *Water Resources Research* 39(2), p 1035.

---

diagnosticPropertyPlot

*Diagnostic Property Plot (base graphics)*

---

**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",
  sort.vars = TRUE
)
```

**Arguments**

f	SoilProfileCollection object
v	character vector of site-level attribute names of logical type
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

<code>dend.label</code>	the name of a site-level attribute (usually unique) annotating dendrogram terminal leaves
<code>sort.vars</code>	sort variables according to natural clustering (TRUE), or use supplied ordering in <code>v</code>

### 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

```
if(require(aqp) &
  require(soilDB) &
  require(latticeExtra)
) {

  # sample data, an SPC
  data(gopheridge, package='soilDB')

  # get depth class
  sdc <- getSoilDepthClass(gopheridge, name = 'hzname')
  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)

}

```

---

diagnosticPropertyPlot2

*Diagnostic Property Plot (lattice)*


---

## Description

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

## Usage

```
diagnosticPropertyPlot2(f, v, k, grid.label = "pedon_id", sort.vars = TRUE)
```

## Arguments

f	SoilProfileCollection object
v	character vector of site-level attribute names of logical type
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
sort.vars	sort variables according to natural clustering (TRUE), or use supplied ordering in v

## 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**

```
if(require(aqp) &
  require(soilDB) &
  require(latticeExtra)
) {

  # sample data, an SPC
  data(gopheridge, package = 'soilDB')

  # get depth class
  sdc <- getSoilDepthClass(gopheridge, name = 'hzname')
  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)

}
```

---

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.order, 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.order	vector of integers that define ordering of coordinates along gradient
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.

**Value**

nothing is returned, function is called to generate graphical output

**Author(s)**

D.E. Beaudette

---

ESS\_by\_Moran\_I                      *Estimate Effective Sample Size*

---

### Description

Estimation of effective sample size (ESS). See Fortin & Dale 2005, p. 223, Equation 5.15 using global Moran's I as 'rho'.

### Usage

ESS\_by\_Moran\_I(n, rho)

### Arguments

n	sample size
rho	Global Moran's I

### Value

numeric; estimated Effective Sample Size

### Author(s)

D.E. Beaudette

### References

Fortin, M.J. and Dale, M.R.T. (2005) Spatial Analysis: A Guide for Ecologists. Cambridge University Press, Cambridge, 1-30.

---

estimateSoilMoistureState

*A very simple estimation of soil moisture state based on volumetric water content*

---

### Description

This is a very simple classification of volumetric water content (VWC) into 5 "moisture states", based on an interpretation of water retention thresholds. Classification is performed using VWC at saturation, field capacity (typically 1/3 bar suction), permanent wilting point (typically 15 bar suction), and water surplus in mm. The inputs to this function are closely aligned with the assumptions and output from `hydromad::hydromad(sma = 'bucket', ...)`.

Soil moisture classification rules are as follows:

- VWC <= pwp: "very dry"

- $VWC > pwp$  AND  $\leq$  (mid-point between  $fc$  and  $pwp$ ): "dry"
- $VWC >$  (mid-point between  $fc$  and  $pwp$ ) AND  $\leq fc$ : "moist"
- $VWC > fc$ : "very moist"
- $VWC > fc$  AND  $U$  (surplus)  $> 4mm$ : "wet"

### Usage

```
estimateSoilMoistureState(
  VWC,
  U,
  sat,
  fc,
  pwp,
  style = c("default", "newhall")
)
```

### Arguments

VWC	vector of volumetric water content (VWC), range is 0-1
U	vector of surplus water (mm)
sat	satiation water content, range is 0-1
fc	field capacity water content, range is 0-1
pwp	permanent wilting point water content, range is 0-1
style	VWC classification style

### Value

vector of moisture states (ordered factor)

### Author(s)

D.E. Beaudette

### Examples

```
# "very moist"
estimateSoilMoistureState(VWC = 0.3, U = 0, sat = 0.35, fc = 0.25, pwp = 0.15)
estimateSoilMoistureState(VWC = 0.3, U = 2, sat = 0.35, fc = 0.25, pwp = 0.15)

#wet
estimateSoilMoistureState(VWC = 0.3, U = 5, sat = 0.35, fc = 0.25, pwp = 0.15)

# "very dry"
estimateSoilMoistureState(VWC = 0.15, U = 0, sat = 0.35, fc = 0.25, pwp = 0.15)

# "dry"
estimateSoilMoistureState(VWC = 0.18, U = 0, sat = 0.35, fc = 0.25, pwp = 0.15)
```



**Description**

Evaluation frost-free days and related metrics from daily climate records.

**Usage**

```
FFD(
  d,
  returnDailyPr = TRUE,
  minDays = 165,
  frostTemp = 32,
  endSpringDOY = 182,
  startFallDOY = 213
)
```

**Arguments**

<code>d</code>	data.frame with columns 'datetime' 'year', and 'value'; 'value' being daily minimum temperature, see details
<code>returnDailyPr</code>	optionally return list with daily summaries
<code>minDays</code>	min number of days of non-NA data in spring   fall, required for a reasonable estimate of FFD
<code>frostTemp</code>	critical temperature that defines "frost" (same units as <code>d\$value</code> )
<code>endSpringDOY</code>	day of year that marks end of "spring" (typically Jan 1 – June 30)
<code>startFallDOY</code>	day of year that marks start of "fall" (typically Aug 1 – Dec 31)

**Details**

The default `frostTemp=32` is suitable for use with minimum daily temperatures in degrees Fahrenheit. Use `frostTemp = 0` for temperatures in degrees Celsius.

[FFD tutorial](#)

**Value**

a data.frame when a `returnDailyPr=FALSE`, otherwise a list with the following elements:

- `summary`: FFD summary statistics as a data.frame
- `fm`: frost matrix
- `Pr.frost`: `Pr(frostlday)`: daily probability of frost

**Author(s)**

D.E. Beaudette

## Examples

```
# 11 years of data from highland meadows
data('HHM', package = 'sharpshootR')
x.ffd <- FFD(HHM, returnDailyPr = FALSE, frostTemp = 32)

str(x.ffd)
```

---

FFDplot

*Plot output from FFD()*

---

## Description

Plot output from FFD()

## Usage

```
FFDplot(s, sub.title = NULL)
```

## Arguments

s	output from <a href="#">FFD</a> , with returnDailyPr = TRUE
sub.title	figure subtitle

## Value

nothing, function is called to generate graphical output

## Examples

```
# 11 years of data from highland meadows
data('HHM', package = 'sharpshootR')
x.ffd <- FFD(HHM, returnDailyPr = TRUE, frostTemp=32)

FFDplot(x.ffd)
```

---

formatPLSS	<i>formatPLSS</i>
------------	-------------------

---

## Description

Format PLSS information into a coded format that can be digested by PLSS web service.

## Usage

```
formatPLSS(p, type = "SN")
```

## Arguments

p	data.frame with chunks of PLSS coordinates
type	an option to format protracted blocks 'PB', unprotracted blocks 'UP', or standard section number 'SN' (default).

## Details

This function is typically accessed as a helper function to prepare data for use within [PLSS2LL](#) function.

## Value

A vector of PLSS codes.

## Note

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

This function requires the following packages: `stringi`.

## Author(s)

D.E. Beaudette, Jay Skovlin, A.G. Brown

## See Also

[PLSS2LL](#)

## Examples

```
# create some data
d <- data.frame(
  id = 1:3,
  qq = c('SW', 'SW', 'SE'),
  q = c('NE', 'NW', 'SE'),
  s = c(17, 32, 30),
```

```

t = c('T36N', 'T35N', 'T35N'),
r = c('R29W', 'R28W', 'R28W'),
type = 'SN',
m = 'MT20',
stringsAsFactors = FALSE
)
# add column names

names(d) <- c('id', 'qq', 'q', 's', 't', 'r', 'type', 'm')
# generate formatted PLSS codes
formatPLSS(d, type='SN')

```

---

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	an sf object, with 1 line segment per feature
precision	digits are rounded to this many places to the right (negative) or left (positive) of the decimal place
algo	hash function algorithm, passed to <code>digest::digest()</code>

### Details

The input sf 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. NA is returned for empty geometries.

### 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

**Examples**

```

if(requireNamespace("sf")) {

# 10 random line segments
# shared end vertices
.x <- runif(n = 11, min = 0, max = 100)
.y <- runif(n = 11, min = 0, max = 100)
m <- matrix(c(.x, .y), ncol = 2, byrow = TRUE)

# init LINESTRING geometries
a <- lapply(1:(nrow(m) - 1), function(i) {
  .idx <- c(i, i+1)
  geom <- sf::st_sfc(sf::st_linestring(m[.idx, ]))
  a <- sf::st_sf(geom)

})

# flatten list -> 10 feature sf object
a <- do.call('rbind', a)

# line hashes
a$id <- generateLineHash(a, precision = 0)

# graphical check
plot(a, lwd = 2, key.width = lcm(4), axes = TRUE, las = 1)

# simulate empty geometry
a$geom[2] <- sf::st_sfc(sf::st_linestring())

# NA returned for empty geometry
generateLineHash(a, precision = 0)

}

```

---

geomorphBySoilSeries-SSURGO

*Geomorphic Position Probability via SDA*


---

**Description**

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

**Usage**

```
hillslopeProbability(s, replaceNA=TRUE)
surfaceShapeProbability(s, replaceNA=TRUE)
geomPosHillProbability(s, replaceNA=TRUE)
geomPosMountainProbability(s, replaceNA=TRUE)
```

**Arguments**

`s` a character vector of soil series names, automatically normalized to upper case

`replaceNA` boolean: should missing classes be converted to probabilities of 0?

**Details**

These functions send a query to the [SDA](http://sdmdataaccess.nrcs.usda.gov/QueryHelp.aspx) 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 at specified geomorphic positions or associated with a surface shape.

**Note**

Probability values are computed from SSURGO data.

**Author(s)**

D.E. Beaudette

---

HenryTimeLine

*Sensor Data Timeline from Henry Mount Soil and Water DB*

---

**Description**

This function generates a simple chart of start/end dates for non-NA sensor data returned by `soilDB::fetchHenry()`. Data are organized according to sensor name + sensor depth.

**Usage**

```
HenryTimeLine(sensor_data, ...)
```

**Arguments**

`sensor_data` `soiltemp`, `soilVWC`, or related data returned by `soilDB::fetchHenry()`

`...` additional arguments to `latticeExtra::segplot`

**Value**

a lattice graphics object

**Author(s)**

D.E. Beaudette

---

HHM

*Highland Meadows*

---

**Description**

11 years of climate data from the Highland Meadows weather station, as maintained by CA DWR.

**Usage**

```
data("HHM")
```

**Format**

A data frame with 3469 observations on the following 12 variables.

station\_id a character vector

dur\_code a character vector

sensor\_num a numeric vector

sensor\_type a character vector

value a numeric vector

flag a character vector

units a character vector

datetime a POSIXct

year a numeric vector

month a factor with levels January February March April May June July August September  
October November December

water\_year a numeric vector

water\_day a numeric vector

---

huePositionPlot

*Hue Position Chart*


---

### Description

A simple visualization of the hue positions for a given Munsell value/chroma according to Soil Survey Technical Note 2.

### Usage

```
huePositionPlot(
  value = 6,
  chroma = 6,
  chip.cex = 4.5,
  label.cex = 0.75,
  contour.dE00 = FALSE,
  origin = NULL,
  origin.cex = 0.75,
  grid.res = 2,
  ...
)
```

### Arguments

value	a single Munsell value
chroma	a single Munsell chroma
chip.cex	scaling for color chip rectangle
label.cex	scaling for color chip
contour.dE00	logical, add dE00 contours from origin, implicitly TRUE when origin is not NULL
origin	point used for distance comparisons can be either single row matrix of CIELAB coordinates, a character vector specifying a Munsell color. By default (NULL) represents CIELAB coordinates (L,0,0), where L is a constant value determined by value and chroma. See examples.
origin.cex	scaling for origin point
grid.res	grid resolution for contours, units are CIELAB A/B coordinates. Caution, small values result in many pair-wise distances which could take a very long time.
...	additional arguments to <a href="#">contour()</a>

### Value

nothing, function is called to generate graphical output



**Examples**

```

huePositionPlot(value = 4, chroma = 4)

huePositionPlot(value = 6, chroma = 6)

huePositionPlot(value = 8, chroma = 8)

huePositionPlot(value = 6, chroma = 6, contour.dE00 = TRUE, grid.res = 2)

# shift origin to arbitrary CIELAB coordinates or Munsell color
huePositionPlot(origin = cbind(40, 5, 15), origin.cex = 0.5)

huePositionPlot(origin = '5G 6/4', origin.cex = 0.5)

huePositionPlot(origin = '10YR 3/4', origin.cex = 0.5)

huePositionPlot(value = 3, chroma = 4, origin = '10YR 3/4', origin.cex = 0.5)

```

---

 hydOrder

*Hydrologic Ordering of a Geomorphic Proportion Matrix*


---

**Description**

Hydrologic Ordering of a Geomorphic Proportion Matrix

**Usage**

```
hydOrder(x, g, clust = TRUE, j.amount = 0)
```

**Arguments**

x	x data.frame, geomorphic proportion matrix, as created by <code>soilDB::fetchOSD(..., extended=TRUE)</code>
g	character, name of geomorphic summary table, one of: <code>c('geomcomp', 'hillpos', 'flats', 'terrace', 'mtnpos', 'shape')</code>
clust	logical, perform clustering of geomorphic proportion matrix
j.amount	amount of noise applied to rows having a duplicate proportion vector, passed to <code>jitter()</code>

**Value**

when `clust = FALSE` a vector of series names, in hydrologic ordering, otherwise a list with the following elements:

- `clust`: rotated `hclust` object
- `hyd.order`: vector of series names, in hydrologic ordering

- `clust.hyd.order`: vector of series names, after clustering + rotation, approximate hydrologic ordering
- `match.rate`: fraction of series matching target hydrologic ordering, after clustering + rotation
- `obj`: objective function value (sum of squared rank differences), used by `iterateHydOrder()`

### Author(s)

D.E. Beaudette

### Examples

```
# example data, similar to results from soilDB::fetchOSD(..., extended = TRUE)
data("OSDexamples")

# no clustering of the geomorphic proportion matrix
h <- hydOrder(OSDexamples$hillpos, g = 'hillpos', clust = FALSE)

# compare with original order

data.frame(
  original = OSDexamples$hillpos$series,
  ordered = h
)

# cluster results
h <- hydOrder(OSDexamples$hillpos, g = 'hillpos', clust = TRUE)
str(h)
```

---

isMineralSoilMaterial *Mineral Soil Material Criteria from 12th Ed. of KST*

---

### Description

Evaluate mineral soil material criteria based on soil organic carbon, clay content, and length of saturation.

### Usage

```
isMineralSoilMaterial(soc, clay, saturation = TRUE)
```

### Arguments

<code>soc</code>	soil organic carbon percent by mass
<code>clay</code>	clay content percent by mass
<code>saturation</code>	logical, cumulative saturation 30+ days

### Value

data.frame of criteria test results

---

iterateHydOrder	<i>Iteratively Attempt Hydrologic Ordering of Geomorphic Proportion Matrix</i>
-----------------	--

---

## Description

Iteratively Attempt Hydrologic Ordering of Geomorphic Proportion Matrix

## Usage

```
iterateHydOrder(
  x,
  g,
  target = 0.9,
  maxIter = 20,
  j.amount = 0.05,
  verbose = FALSE,
  trace = FALSE
)
```

## Arguments

x	data.frame geomorphic proportion matrix, as created by <code>soilDB::fetchOSD(..., extended=TRUE)</code>
g	name of geomorphic summary table, one of: <code>c('geomcomp', 'hillpos', 'flats', 'terrace', 'mtnpos', 'shape')</code>
target	numeric, target match rate
maxIter	integer, maximum number of perturbations of geomorphic probability matrix
j.amount	numeric, amount of noise applied to rows with too few unique values, passed to <code>jitter()</code>
verbose	logical, additional output printed via message
trace	logical, additional list of results for each iteration

## Details

This function is used by the suite of geomorphic proportion visualization functions (`viz*`) to attempt rotation of a dendrogram according to "hydrologic ordering" rules. A perfect rotation is not always possible, and reported as a match rate in the returned score value

## Value

A list with the following elements:

- `clust`: rotated `hclust` object
- `hyd.order`: vector of series names, in hydrologic ordering

- `clust.hyd.order`: vector of series names, after clustering + rotation, approximate hydrologic ordering
- `match.rate`: fraction of series matching target hydrologic ordering, after clustering + rotation
- `obj`: objective function value (sum of squared rank differences), used by `iterateHydOrder()`
- `niter`: number of iterations
- `trace`: list of results by iteration, only when `trace = TRUE`

### Author(s)

D.E. Beaudette

### Examples

```
# example data, similar to results from soilDB::fetchOSD(..., extended = TRUE)
data("OSDexamples")

# single iteration of hydrologic ordering
h1 <- hydOrder(OSDexamples$hillpos, g = 'hillpos', clust = TRUE)

# perform several iterations, keep the best one
h2 <- iterateHydOrder(OSDexamples$hillpos, 'hillpos', verbose = TRUE)

# compare: only slightly better match rate achieved
h1$match.rate
h2$match.rate

# return trace log for eval of objective function
# increase max iterations
h2 <- iterateHydOrder(OSDexamples$hillpos, 'hillpos', maxIter = 100, verbose = TRUE, trace = TRUE)

# inspect objective function evolution
tr <- h2$trace
obj <- sapply(tr, '[[', 'obj')

plot(obj, type = 'b')
hist(obj)

# in this case the clustering of hillpos proportions has only two possible configurations
```

---

joinAdjacency

*Join Document Adjacency*

---

### 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** data.frame or similar object, each row represents a single shared edge (typically sf LINESTRING feature)

**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\(\)](#)

---

 LL2PLSS

*LL2PLSS*


---

**Description**

Uses latitude and longitude coordinates to return the PLSS section geometry from the BLM PLSS web service.

**Usage**

```
LL2PLSS(x, y, returnlevel = c("I", "S"))
```

**Arguments**

**x** longitude coordinates (WGS84)

**y** latitude coordinates (WGS84)

**returnlevel** 'S' for "Section" or 'I' for "Intersection" (subsections)

**Details**

This function accepts geographic coordinates and returns the PLSS fabric geometry to the quarter-quarter section. `returnlevel` defaults to 'I' which returns smallest intersected sectional aliquot geometry, 'S' will return the section geometry of the coordinates. See [https://gis.blm.gov/arcgis/rest/services/Cadastral/BLM\\_N](https://gis.blm.gov/arcgis/rest/services/Cadastral/BLM_N) for details.

**Value**

sf object with geometry and PLSS definition.

**Note**

This function requires the following packages: httr, jsonlite, and sp.

**Author(s)**

D.E. Beaudette, Jay Skovlin, A.G. Brown

**See Also**

[PLSS2LL](#), [formatPLSS](#)

---

moistureStateProportions

*Compute moisture state proportions*

---

**Description**

Compute moisture state proportions

**Usage**

```
moistureStateProportions(x, id = "compname", step = c("month", "week", "doy"))
```

**Arguments**

x	data.frame created by <a href="#">dailyWB()</a> or <a href="#">dailyWB_SSURGO()</a>
id	character, column name identifying sites, components, or soil series
step	time step, one of 'month', 'week', or 'doy'

**Value**

data.frame

---

moistureStateStats	<i>Statistics on Soil Moisture State</i>
--------------------	--

---

**Description**

Statistics on Soil Moisture State

**Usage**

```
moistureStateStats(x, id = "compname")
```

**Arguments**

x	data.frame, created by <a href="#">moistureStateProportions()</a>
id	name of ID column

**Value**

data.frame containing the most-likely moisture state and Shannon entropy.

---

moistureStateThreshold	<i>Apply a threshold to soil moisture states</i>
------------------------	--

---

**Description**

Apply a threshold to soil moisture states

**Usage**

```
moistureStateThreshold(
  x,
  id = "compname",
  threshold = "moist",
  operator = c("<", ">", "==", "<=", ">=")
)
```

**Arguments**

x	a data.frame created by <a href="#">dailyWB()</a> or <a href="#">dailyWB_SSURGO()</a>
id	character, column name identifying sites, soils, or soil series
threshold	moisture state threshold, see <a href="#">estimateSoilMoistureState</a>
operator	one of "<", ">", "==", "<=", or ">="

**Value**

data.frame

**Author(s)**

D.E. Beaudette

---

monthlyWB

*Monthly Water Balances*

---

**Description**

Perform a monthly water balance by "leaky bucket" model, inspired by code from `bucket.sim` of `hydromad` package, as defined in Bai et al., (2009) (model "SMA\_S1"). The plant available water-holding storage (soil thickness \* awc) is used as the "bucket capacity". All water in excess of this capacity is lumped into a single "surplus" term.

**Usage**

```
monthlyWB(
  AWC,
  PPT,
  PET,
  S_init = 1,
  starting_month = 1,
  rep = 1,
  keep_last = FALSE
)

monthlyWB_summary(w)
```

**Arguments**

AWC	available water-holding capacity (mm), typically thickness (mm) * awc (fraction)
PPT	time-series of monthly PPT (mm), calendar year ordering
PET	time-series of monthly PET (mm), calendar year ordering
S_init	initial fraction of AWC filled with water (values 0-1)
starting_month	starting month index, 1=January, 9=September
rep	number of cycles to run water balance
keep_last	keep only the last iteration of the water balance
w	used for <code>monthlyWB_summary()</code> : a data.frame, such as result of <code>monthlyWB()</code> ;



## Details

See the [monthly water balance tutorial](#) for further examples and discussion.

A number of important assumptions are made by this style of water balance modeling:

- the concept of field capacity is built into the specified bucket size
- the influence of aquitards or local terrain cannot be integrated into this model
- interception is not used in this model

## Value

a `data.frame` with the following elements:

- PPT: monthly PPT (mm)
- PET: monthly PET (mm)
- U: monthly surplus (mm)
- S: monthly soil moisture storage (mm)
- ET: monthly AET (mm)
- D: monthly deficit (mm)
- month: month number
- mo: month label

`monthlyWB_summary()`: a `data.frame` containing:

- cumulative (dry, moist, wet) days
- consecutive (dry\_con, moist\_con, wet\_con) days
- total deficit (total\_deficit) in mm
- total surplus (total\_surplus) in mm
- total actual evapotranspiration (total\_AET) in mm
- annual actual evapotranspiration to potential evapotranspiration ratio (annual\_AET\_PET\_ratio)

## References

Arkley R, Ulrich R. 1962. The use of calculated actual and potential evapotranspiration for estimating potential plant growth. *Hilgardia* 32(10):443-469.

Bai, Y., T. Wagener, P. Reed (2009). A top-down framework for watershed model evaluation and selection under uncertainty. *Environmental Modelling and Software* 24(8), pp. 901-916.

Farmer, D., M. Sivapalan, Farmer, D. (2003). Climate, soil and vegetation controls upon the variability of water balance in temperate and semiarid landscapes: downward approach to water balance analysis. *Water Resources Research* 39(2), p 1035.

---

Moran\_I\_ByRaster

*Compute Moran's I for a raster sampled from a mapunit extent*


---

### Description

Compute Moran's I using a subset of sample collected within the extent of a mapunit. This is likely an under-estimate of SA because we are including pixels both inside/outside MU delineations

### Usage

```
Moran_I_ByRaster(
  r,
  mu.extent = NULL,
  n = NULL,
  k = NULL,
  do.correlogram = FALSE,
  cor.order = 5,
  crop.raster = TRUE
)
```

### Arguments

<code>r</code>	single SpatRaster
<code>mu.extent</code>	SpatVector representation of mapunit polygons bounding box (via <code>terra::ext()</code> )
<code>n</code>	number of regular samples (what is a reasonable value?)
<code>k</code>	number of neighbors used for weights matrix
<code>do.correlogram</code>	compute correlogram?
<code>cor.order</code>	order of correlogram
<code>crop.raster</code>	optionally disable cropping of the raster layer

### Details

This function uses the `spdep::moran.test()` function

### Value

If `do.correlogram` is TRUE a list with estimated Moran's I (`$I`) and the correlogram (`$correlogram`), otherwise the estimated Moran's I value.

### Author(s)

D.E. Beaudette

---

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 <code>SoilProfileCollection</code> object
v	the name of a site-level attribute that is a factor, or can be coerced to 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

```
if(require(soilDB) &
  require(aqp) &
  require(latticeExtra)) {

  # sample data, an SPC
  data(loafercreek, package='soilDB')

  # convert to logical matrix
  hp <- multinomial2logical(loafercreek, 'hillslopeprof')

  # 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 = 'bedrckkind', dend.label = 'pedon_id')
}

```

---

OSDexamples

*Example output from soilDB::fetchOSD()*


---

### Description

These example data are used to test various functions in this package when network access may be limited.

### Usage

```
data(OSDexamples)
```

### Format

An object of class `list` of length 17.

---

PCP\_plot

*Percentiles of Cumulative Precipitation*


---

### Description

Generate a plot representing percentiles of cumulative precipitation, given a historic record, and criteria for selecting a year of data for comparison.

### Usage

```

PCP_plot(
  x,
  this.year,
  this.day = NULL,
  method = "exemplar",
  q.color = "RoyalBlue",

```

```

    c.color = "firebrick",
    ...
)

```

### Arguments

x	result from CDECquery for now, will need to generalize to other sources
this.year	a single water year, e.g. 2020
this.day	optional integer representing days since start of selected water year
method	'exemplar' or 'daily', currently 'exemplar' is the only method available
q.color	color of percentiles cumulative precipitation
c.color	color of selected year
...	additional arguments to plot

### Details

This is very much a work in progress. Further examples at <https://ncss-tech.github.io/AQP/sharpsshootR/CDEC.html>, and <https://ncss-tech.github.io/AQP/sharpsshootR/cumulative-PPT.html>.

### Value

nothing, this function is called to create graphical output

### Author(s)

D.E. Beaudette

### See Also

[waterDayYear](#)

---

percentileDemo

*Demonstration of Percentiles vs. Mean / SD*

---

### Description

This function can be used to graphically demonstrate the relationship between distribution shape, an idealized normal distribution (based on sample mean and sd) shape, and measures of central tendency / spread.

### Usage

```

percentileDemo(x, labels.signif = 3, pctile.color = "RoyalBlue",
mean.color = "Orange", range.color = "DarkRed",
hist.breaks = 30, boxp = FALSE, ...)

```

**Arguments**

x	vector of values to summarize
labels.signif	integer, number of significant digits to be used in figure annotation
pctile.color	color used to demonstrate range from 10th to 90th percentiles
mean.color	color used to specify mean +/- 2SD
range.color	color used to specify data range
hist.breaks	integer, number of suggested breaks to hist
boxp	logical, add a box and whisker plot?
...	further arguments to plot

**Value**

A 1-row matrix of summary stats is invisibly returned.

**Note**

This function is mainly for educational purposes.

**Author(s)**

D.E. Beaudette

**References**

<https://ncss-tech.github.io/soil-range-in-characteristics/why-percentiles.html>

**Examples**

```
x <- rnorm(100)
percentileDemo(x)
```

```
x <- rlnorm(100)
percentileDemo(x)
```

---

plotAvailWater

*Visual Demonstration of Available Soil Water*

---

**Description**

Generate a simplistic diagram of the various fractions of water held within soil pore-space. Largely inspired by [Figure 2 from O'Geen \(2013\)](#).

**Usage**

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

**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
annotate	logical, annotate AWC

**Value**

nothing, function is called to generate graphical output

**Author(s)**

D.E. Beaudette

**References**

O'Geen, A. T. (2013) Soil Water Dynamics. Nature Education Knowledge 4(5):9.

**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)
```

```

if(requireNamespace("aqp")) {

  # demonstration using idealized AWC by soil texture
  data("ROSETTA.centroids", package = "aqp")

  # subset columns
  x <- ROSETTA.centroids[, c('texture', 'pwp', 'fc', 'sat', 'awc')]

  # adjust to expected names / additional data required by plotAvailWater
  names(x)[1] <- 'name'
  x$solid <- with(x, 1 - sat)

  # re-order based on approximate AWC
  x <- x[order(x$awc), ]

  op <- par(no.readonly = TRUE)

  par(mar=c(5, 6.5, 0.5, 0.5))
  plotAvailWater(x, name.cex = 1)

  par(op)

}

# use some real data from SSURGO
if(requireNamespace("curl") &
  curl::has_internet() &
  require(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)

  op <- par(no.readonly = TRUE)

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

  par(op)

}

```



---

 plotGeomorphCrossSection

*Present a SoilProfileCollection aligned to a geomorphic summary as cross-section.*

---

### Description

Present a SoilProfileCollection aligned to a geomorphic summary as cross-section.

### Usage

```
plotGeomorphCrossSection(
  x,
  type = c("line", "bar"),
  g = "hillpos",
  clust = TRUE,
  col = c("#377EB8", "#4DAF4A", "#984EA3", "#FF7F00", "#E41A1C"),
  ...
)
```

### Arguments

x	resulting list from <code>soilDB::fetchOSD(..., extended = TRUE)</code>
type	character, 'line' for line plot or 'bar' for barplot of geomorphic proportions
g	character, select a geomorphic summary. Currently 'hillpos' (2D hillslope position) is the only supported choice.
clust	logical, use clustering order of geomorphic proportions or exact hydrologic ordering, see <a href="#">hydOrder()</a>
col	character vector of colors
...	additional arguments to <a href="#">iterateHydOrder()</a>

### Author(s)

D.E. Beaudette

---

 plotProfileDendrogram *Plot soil profiles below a dendrogram*


---

### Description

Plot soil profiles below a dendrogram

**Usage**

```
plotProfileDendrogram(
  x,
  clust,
  rotateToProfileID = FALSE,
  scaling.factor = 0.01,
  width = 0.1,
  y.offset = 0.1,
  dend.y.scale = max(clust$height * 2, na.rm = TRUE),
  dend.color = par("fg"),
  dend.width = 1,
  dend.type = c("phylogram", "cladogram"),
  debug = FALSE,
  ...
)
```

**Arguments**

<code>x</code>	a <code>SoilProfileCollection</code> object
<code>clust</code>	a hierarchical clustering object generated by <code>hclust</code> , <code>cluster::agnes</code> , or <code>cluster::diana</code>
<code>rotateToProfileID</code>	logical, attempt rotation of dendrogram according to original profile IDs, requires <code>dendExtend</code> package
<code>scaling.factor</code>	vertical scaling of the profile heights (may have to tinker with this)
<code>width</code>	scaling of profile widths
<code>y.offset</code>	vertical offset for top of profiles
<code>dend.y.scale</code>	extent of y-axis (may have to tinker with this)
<code>dend.color</code>	dendrogram line color
<code>dend.width</code>	dendrogram line width
<code>dend.type</code>	dendrogram type, passed to <code>plot.phylo()</code> , either "phylogram" or "cladogram"
<code>debug</code>	logical, optionally print debugging data
<code>...</code>	additional arguments to <code>plotSPC</code>

**Details**

This function places soil profile sketches below a dendrogram.

**Value**

a `data.frame` of IDs and linking structure

**Note**

You may have to tinker with some of the arguments to get optimal arrangement and scaling of soil profiles.

**Author(s)**

D.E. Beaudette

---

`plotSoilRelationChordGraph`*Visualize Soil Relationships via Chord Diagram*

---

**Description**

Visualize Soil Relationships via Chord Diagram

**Usage**

```
plotSoilRelationChordGraph(  
  m,  
  s,  
  mult = 2,  
  base.color = "grey",  
  highlight.colors = c("RoyalBlue", "DarkOrange", "DarkGreen"),  
  add.legend = TRUE,  
  ...  
)
```

**Arguments**

<code>m</code>	an adjacency matrix, no NA allowed
<code>s</code>	soil of interest, must exist in the column or row names of <code>m</code>
<code>mult</code>	multiplier used to re-scale data in <code>m</code> associated with <code>s</code>
<code>base.color</code>	color for all soils other than <code>s</code> and 1st and 2nd most commonly co-occurring soils
<code>highlight.colors</code>	vector of 3 colors: soil of interest, 1st most common, 2nd most common
<code>add.legend</code>	logical, add a legend
<code>...</code>	additional arguments passed to <code>circlize::chordDiagramFromMatrix</code>

**Details**

This function is experimental. Documentation pending. See <http://jokergoo.github.io/circlize/> for ideas.

**Value**

nothing, function is called to generate graphical output

**Author(s)**

D.E. Beaudette

---

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 = c("network", "dendrogram", "none"),
  graph.mode = "upper",
  spanning.tree = NULL,
  del.edges = NULL,
  vertex.scaling.method = "degree",
  vertex.scaling.factor = 2,
  edge.scaling.factor = 1,
  vertex.alpha = 0.65,
  edge.transparency = 1,
  edge.col = grey(0.5),
  edge.highlight.col = "royalblue",
  g.layout = igraph::layout_with_fr,
  vertex.label.color = "black",
  delete.singletons = FALSE,
  ...
)
```

### Arguments

<code>m</code>	adjacency matrix
<code>s</code>	central component; an empty character string is interpreted as no central component
<code>plot.style</code>	plot style ('network', or 'dendrogram'), or 'none' for no graphical output
<code>graph.mode</code>	interpretation of adjacency matrix: 'upper' or 'directed', see details
<code>spanning.tree</code>	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 <code>spanning.tree</code> . See details and examples.
<code>del.edges</code>	optionally delete edges with weights less than the specified quantile (0-1)
<code>vertex.scaling.method</code>	'degree' (default) or 'distance', see details
<code>vertex.scaling.factor</code>	scaling factor applied to vertex size

edge.scaling.factor	optional scaling factor applied to edge width
vertex.alpha	optional transparency setting for vertices (0-1)
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 igraph::layout_with_fr
vertex.label.color	vertex label color
delete.singletons	optionally delete vertices with no edges (degree == 0)
...	further arguments passed to plotting function

### Details

Vertex size is based on a normalized index of connectivity:

- "degree" size =  $\sqrt{\text{igraph}::\text{degree}(g) / \max(\text{igraph}::\text{degree}(g))} * \text{scaling.factor}$
- "distance" size =  $\sqrt{\text{igraph}::\text{distance}(V \rightarrow s) / \max(\text{igraph}::\text{distance}(V \rightarrow s))} * \text{scaling.factor}$ , where  $\text{distance}(V \rightarrow s)$  is the distance from all nodes to the named series, s.

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' auxiliary edges—defined by a value between 0 and 1.

The `graph.mode` argument is passed to `igraph::graph_from_adjacency_matrix()` and determines how vertex relationships are coded in the adjacency matrix `m`. Typically, the default value of 'upper' (the upper triangle of `m` contains adjacency information) is the desired mode. If `m` contains directional information, set `graph.mode` to 'directed'. This has the side-effect of altering the default community detection algorithm from `igraph::cluster_fast_greedy` to `igraph::cluster_walktrap`.

### Value

an igraph graph object is invisibly returned

### Note

This function is a work in progress, ideas welcome.

### Author(s)

D.E. Beaudette

**Examples**

```

if (requireNamespace("igraph")) {
  # 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')

  op <- par(no.readonly = TRUE)

  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 pctile
  plotSoilRelationGraph(m, spanning.tree=0.75)

  par(op)

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

    # get similar data from soilweb, for the Pardee series
    s <- 'pardee'
    d <- siblings(s, component.data = TRUE)

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

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

    # build adj. matrix and plot
    m <- component.adj.matrix(d$sib.data)
    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)
  }
}

```

---

plotTransect

*Arrange Profiles along a Transect*


---

### Description

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

### Usage

```

plotTransect(
  s,
  xy,
  grad.var.name,
  grad.var.order = order(site(s)[[grad.var.name]]),
  transect.col = "RoyalBlue",
  tick.number = 7,
  y.offset = 100,
  scaling.factor = 0.5,
  distance.axis.title = "Distance Along Transect (km)",
  grad.axis.title = NULL,
  dist.scaling.factor = 1000,
  spacing = c("regular", "relative"),
  fix.relative.pos = list(thresh = 0.6, maxIter = 5000),
  ...
)

```

### Arguments

s	SoilProfileCollection object
xy	sf object, defining point coordinates of soil profiles, must be in same order as s, must be a projected coordinate reference system (UTM, AEA, etc.)
grad.var.name	the name of a site-level attribute containing gradient values
grad.var.order	optional indexing vector used to override sorting along grad.var.name
transect.col	color used to plot gradient (transect) values
tick.number	number of desired ticks and labels on the gradient axis
y.offset	vertical offset used to position profile sketches
scaling.factor	scaling factor applied to profile sketches

```

distance.axis.title      a title for the along-transect distances
grad.axis.title         a title for the gradient axis
dist.scaling.factor     scaling factor (divisor) applied to linear distance units, default is conversion
                        from m to km (1000)
spacing                 profile sketch spacing style: "regular" (profiles aligned to an integer grid) or
                        "relative" (relative distance along transect)
fix.relative.pos        adjust relative positions in the presence of overlap, FALSE to suppress, otherwise
                        list of arguments to aqp::fixOverlap
...                    further arguments passed to aqp::plotSPC.

```

### 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

```

if(require(aqp) &
require(sf) &
require(soilDB)
) {

```



```
library(aqp)
library(soilDB)
library(sf)

# sample data
data("mineralKing", package = "soilDB")

# device options are modified locally, reset when done
op <- par(no.readonly = TRUE)

# quick overview
par(mar=c(1,1,2,1))
groupedProfilePlot(mineralKing, groups='taxonname', print.id=FALSE)

# setup point locations
s <- site(mineralKing)
xy <- st_as_sf(s, coords = c('x_std', 'y_std'))
st_crs(xy) <- 4326

# convert to suitable projected CRS
# projected CRS, UTM z11 NAD83 (https://epsg.io/26911)
xy <- st_transform(xy, 26911)

# adjust margins
par(mar = c(4.5, 4, 4, 1))

# standard transect plot, profile sketches arranged along integer sequence
plotTransect(mineralKing, xy, grad.var.name = 'elev_field',
             grad.axis.title = 'Elevation (m)', label = 'pedon_id', name = 'hzname')

# default behavior, attempt adjustments to prevent over-plot and preserve relative spacing
# use set.seed() to fix outcome
plotTransect(mineralKing, xy, grad.var.name = 'elev_field',
             grad.axis.title = 'Elevation (m)', label = 'pedon_id',
             name = 'hzname', width = 0.15, spacing = 'relative')

# attempt relative positioning based on scaled distances, no corrections for overlap
# profiles are clustered in space and therefore over-plot
plotTransect(mineralKing, xy, grad.var.name = 'elev_field',
             grad.axis.title = 'Elevation (m)', label = 'pedon_id', name = 'hzname',
             width = 0.15, spacing = 'relative', fix.relative.pos = FALSE)

# customize arguments to aqp::fixOverlap()
plotTransect(mineralKing, xy, grad.var.name = 'elev_field', crs = crs.utm,
             grad.axis.title = 'Elevation (m)', label = 'pedon_id', name = 'hzname',
             width = 0.15, spacing = 'relative',
             fix.relative.pos = list(maxIter=6000, adj=0.2, thresh=0.7))

plotTransect(mineralKing, xy, grad.var.name = 'elev_field', crs = crs.utm,
             grad.axis.title = 'Elevation (m)', label = 'pedon_id', name = 'hzname',
             width = 0.2, spacing = 'relative',
```



AWC	available water-holding capacity (mm), typically the value used in monthlyWB() and stored as an attribute of WB
sw.col	color for soil water ("storage)
surplus.col	color for surplus water
et.col	color for ET
deficit.col	color for deficit
pch	plotting character for PPT and PET points
pt.cex	character expansion factor for PPT and PET points
pt.col	point symbol color for PPT and PET points
pt.bg	point symbol background color for PPT and PET points
lty	line type for PPT and PET lines (c(1, 2))
lwd	line width for PPT and PET curves
n.ticks	approximate number of tick marks on positive and negative y-axis
grid.col	horizontal grid line color
month.cex	scaling factor for month labels (x-axis)
legend.cex	scaling factor for legend
ylim	optional vector of y-axis limits, c(-min, max), typically used when comparing drastically different water balances in the same figure. Default limits are usually best for a single water balance plot.

**Value**

nothing, function is called to generate graphical output

**Note**

You may have to adjust figure margins and size to get all of the elements to "look right".

**Author(s)**

D.E. Beaudette and J.M. Skovlin

**Examples**

```

if(requireNamespace('hydromad')) {

## A shallow / droughty soil near Sonora CA
# 100mm (4") AWC
AWC <- 100
PPT <- c(171, 151, 138, 71, 36, 7, 1, 2, 11, 48, 102, 145)
PET <- c(15.17, 18.26, 30.57, 42.95, 75.37, 108.05, 139.74, 128.9, 93.99, 59.84, 26.95, 14.2)

# water-year
# three years
x.wb <- monthlyWB(AWC, PPT, PET, S_init = 0, starting_month = 9, rep = 3)

```

```

x.wb[x.wb$mo == 'Sep', ]

# plot all three years
plotWB(x.wb)

# water-year / last iteration
x.wb <- monthlyWB(AWC, PPT, PET, S_init = 0,
                  starting_month = 9, rep = 3,
                  keep_last = TRUE
)

# plot
plotWB(x.wb)

## Drummer series (Fine-silty, mixed, superactive, mesic Typic Endoaquolls), southern IL

AWC <- 244
PPT <- c(36, 37, 54, 82, 98, 96, 92, 75, 69, 70, 65, 50)
PET <- c(0, 0, 12, 46, 90, 130, 145, 128, 88, 46, 14, 0)

# using calendar year
x.wb <- monthlyWB(AWC, PPT, PET, S_init = 0,
                  starting_month = 1, rep = 3,
                  keep_last = TRUE
)

plotWB(x.wb)

}

```

---

plotWB\_lines

---

*Line / Area Visualization for Monthly Water Balance*


---

### Description

Pending.

### Usage

```

plotWB_lines(
  WB,
  cols = c("#759CC9", "#EB6D6E", "#7FC47D"),
  line.col = "black",
  line.lty = c(1, 2, 3),
  interpolator = c("spline", "linear"),
  spline.method = c("natural", "periodic"),
  month.cex = 1,
  legend.cex = 0.9
)

```

**Arguments**

WB	output from <code>monthlyWB()</code>
cols	vector of three colors used for area under PPT, PET, and AET curves
line.col	single color used for PPT, PET, and AET lines
line.lty	vector of three line styles used for PPT, PET, AET curves
interpolator	spline or linear interpolation of monthly values, use of spline may lead to minor smoothing artifacts in shaded areas
spline.method	when <code>interpolator = 'spline'</code> , argument passed to <code>splinefun(..., method = spline.method)</code>
month.cex	scaling factor for month labels
legend.cex	scaling factor for legend

**Value**

nothing, function is called to generate graphical output

**Author(s)**

J.M. Skovlin and D.E. Beaudette

**Examples**

```
if(requireNamespace('hydromad')) {  
  
  ## A shallow / droughty soil near Sonora CA  
  # 100mm (4") AWC  
  AWC <- 100  
  PPT <- c(171, 151, 138, 71, 36, 7, 1, 2, 11, 48, 102, 145)  
  PET <- c(15.17, 18.26, 30.57, 42.95, 75.37, 108.05, 139.74, 128.9, 93.99, 59.84, 26.95, 14.2)  
  
  # calendar-year  
  # three year warm-up  
  x.wb <- monthlyWB(AWC, PPT, PET, S_init = 0, starting_month = 1, rep = 3, keep_last = TRUE)  
  
  # plot  
  plotWB_lines(x.wb)  
  
}
```

---

 PLSS2LL

*PLSS2LL*


---

**Description**

Fetch latitude and longitude (centroid) coordinates for coded PLSS information from the BLM PLSS web service.

**Usage**

```
PLSS2LL(p, plssid = "plssid")
```

**Arguments**

p	data.frame with chunks of PLSS definition
plssid	column name containing PLSS ID

**Value**

A data.frame of PLSS codes and coordinates.

**Note**

This function expects that the dataframe will have a 'plssid' column generated by the formatPLSS function. Requires the following packages: httr, and jsonlite.

**Author(s)**

D.E. Beaudette, Jay Skovlin, A.G. Brown

**See Also**

[LL2PLSS](#), [formatPLSS](#)

---

 polygonAdjacency

*Summarize Spatial Adjacency of Polygon Fabric*


---

**Description**

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

**Usage**

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

**Arguments**

x	sf object containing simple polygon features, some of which should share edges
v	character, name of column in attribute table describing map unit labels
...	additional arguments passed to <code>spdep::poly2nb()</code>

**Details**

Examples are presented in [this tutorial](#).

**Value**

a list containing:

- `commonLines`: an integer vector of feature IDs, describing polygons sharing edges and values of `v` (map unit labels)
- `adjMat`: weighted adjacency matrix, suitable for visualization with `plotSoilRelationGraph()`

**Author(s)**

D.E. Beaudette

---

```
prepareDailyClimateData
```

*Prepare daily climate data (DAYMET) for a single point*

---

**Description**

This function returns daily climate data required for a simple water balance (and more), using three packages:

- `elevatr`: elevation data at `x`
- `daymetr`: DAYMET data at `x` for years `start` through `end`
- `Evapotranspiration`: Makkink formulation for estimating reference crop evapotranspiration

**Usage**

```
prepareDailyClimateData(x, start, end, onlyWB = TRUE)
```

**Arguments**

x	sf object representing a single point
start	start year (1998)
end	end year (2018)
onlyWB	logical, return just those date required by <code>dailyWB</code>

**Value**

a `data.frame`

---

```
prepare_SSURGO_hydro_data
```

*Get and prepare basic soil hydraulic parameters from SSURGO via SDA*

---

### Description

Get and prepare basic soil hydraulic parameters from SSURGO via SDA

### Usage

```
prepare_SSURGO_hydro_data(cokeys, max.depth)
```

### Arguments

cokeys	vector of component keys (cokey) in current SSURGO snapshot
max.depth	target depth of aggregation (cm), corrected later by real soil depth as reported by <code>slab()</code>

### Details

Weighted mean soil hydraulic parameters are returned over the interval of  $0$ –`max.depth`, calculated by `aqp::slab()`.

### Value

a list containing:

- SPC: SoilProfileCollection
- agg: aggregate representation of hydraulic parameters, by cokey

### Author(s)

D.E. Beaudette

---

```
reconcileOSDGeomorph
```

*Reconcile IDs between a SPC and associated geomorphic proportion table*

---

### Description

This function can assist with linked visualizations that include soil morphology data stored in a SoilProfileCollection and geomorphic proportions stored in a data.frame, as returned by `soilDB::fetchOSD()`.



**Usage**

```
reconcileOSDGeomorph(
  x,
  selection = c("hillpos", "geomcomp", "flats", "mtnpos", "terrace", "shape_across",
               "shape_down")
)
```

**Arguments**

x                    resulting list from soilDB::fetchOSD(..., extended = TRUE)  
 selection           character, name of geomorphic proportion table

**Value**

a list with subset SoilProfileCollection and data.frame of geomorphic proportions, selection is preserved as an attribute.

**Author(s)**

D.E. Beaudette

---

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='regular', p4s=NULL)
```

**Arguments**

p                    a Polygon object, with coordinates in a projected CRS with units of meters  
 n.pts.per.ac        requested sampling density in points per acre (results will be close)  
 min.samples        minimum requested number of samples per polygon  
 sampling.type      sampling type  
 p4s                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. Invalid geometries may cause errors or yield incorrect sample sizes.

**Author(s)**

D.E. Beaudette

**See Also**

[constantDensitySampling](#)

---

sampleRasterStackByMU *Sample a Raster Stack*

---

**Description**

Sample a raster stack by map unit polygons, at a constant density.

**Usage**

```
sampleRasterStackByMU(
  mu,
  mu.set,
  mu.col,
  raster.list,
  pts.per.acre,
  p = c(0, 0.05, 0.25, 0.5, 0.75, 0.95, 1),
  progress = TRUE,
  estimateEffectiveSampleSize = TRUE,
  polygon.id = "pID"
)
```

**Arguments**

mu	a SpatialPolygonsDataFrame object in a projected coordinate reference system (CRS)
mu.set	character vector of map unit labels to be sampled
mu.col	column name in attribute table containing map unit labels
raster.list	a list containing raster names and paths, see details below
pts.per.acre	target sampling density in points per acre
p	percentiles for polygon area stats, e.g. c(0.05, 0.25, 0.5, 0.75, 0.95)

progress	logical, print a progress bar while sampling?
estimateEffectiveSampleSize	estimate an effective sample size via Moran's I?
polygon.id	Column name containing unique polygon IDs; default: "pID"; calculated if missing

### Details

This function is used by various NRCS reports that summarize or compare concepts defined by collections of polygons using raster data sampled from within each polygon, at a constant sampling density. Even though the function name includes "RasterStack", this function doesn't actually operate on the "stack" object as defined in the raster package. The collection of raster data defined in `raster.list` do not have to share a common coordinate reference system, grid spacing, or extent. Point samples generated from `mu` are automatically converted to the CRS of each raster before extracting values. The extent of each raster in `raster.list` must completely contain the extent of `mu`.

### Value

A list containing:

`raster.samples` a data.frame containing samples from all rasters in the stack  
`area.stats` a data.frame containing area statistics for all map units in the collection  
`unsampled.ids` an index to rows in the original SPDF associated with polygons not sampled  
`raster.summary` a data.frame containing information on sampled rasters  
`Moran_I` a data.frame containing estimates Moran's I (index of spatial autocorrelation)

### Author(s)

D.E. Beaudette

### See Also

[constantDensitySampling](#), [sample.by.poly](#)

---

samplingStability      *Estimate Sampling Stability*

---

### Description

Stability is defined as the width of the 5th-95th percentile range, over `n.reps` replications of median estimates associated with sampling events. The resulting width is scaled by the population median and returned as a fraction.

**Usage**

```
samplingStability(
  mu,
  r,
  n.set = c(0.01, 0.1, 0.5, 1, 2),
  n.reps = 10,
  p.id = "pID"
)
```

**Arguments**

mu	map unit polygons, must have polygon ID, must be in CRS with units of meters
r	SpatRaster
n.set	set of sampling density values to try
n.reps	number of replications
p.id	polygon ID column name

**Value**

data.frame with median stability values as percentage of population median, range: [0, 1]

**Author(s)**

D.E. Beaudette

---

simpleWB

*Simple interface to the hydromad "leaky bucket" soil moisture model*

---

**Description**

Simple interface to the hydromad "leaky bucket" soil moisture model.

**Usage**

```
simpleWB(
  PPT,
  PET,
  D,
  thickness,
  sat,
  fc,
  pwp,
  S_0 = 0.5,
  a.ss = 0.05,
  M = 0,
  etmult = 1
)
```

**Arguments**

PPT	precipitation series (mm)
PET	potential ET series (mm)
D	dates
thickness	soil thickness (cm)
sat	volumetric water content at saturation (satiated water content)
fc	volumetric water content at field capacity (typically 1/3 bar suction)
pwp	volumetric water content at permanent wilting point (typically 15 bar suction)
S_0	initial soil moisture as a fraction of total water storage (mm)
a.ss	recession coefficients for subsurface flow from saturated zone, should be > 0
M	fraction of area covered by deep-rooted vegetation
etmult	multiplier for PET

**Details**

Adjustments for coarse fragments should be made by reducing thickness.

**Value**

a data.frame

**References**

Farmer, D., M. Sivapalan, Farmer, D. (2003). Climate, soil and vegetation controls upon the variability of water balance in temperate and semiarid landscapes: downward approach to water balance analysis. *Water Resources Research* 39(2), p 1035.

Bai, Y., T. Wagener, P. Reed (2009). A top-down framework for watershed model evaluation and selection under uncertainty. *Environmental Modelling and Software* 24(8), pp. 901-916.

---

site_photos_kml	<i>site_photos_kml</i>
-----------------	------------------------

---

**Description**

Generates a KML file of site locations with associated site photos and a link to a pedon description report.

**Usage**

```
site_photos_kml(data,
filename='photos.kml', make.image.grid=FALSE,
file.source = c('local', 'relative')
)
```

**Arguments**

data	a dataframe
filename	full file path and name with .kml extension
make.image.grid	logical, include linked site images, default is FALSE
file.source	'local' sources the image files to a specific system path, 'relative' sources the image files to files folder that can be included and referenced within a .kmz file

**Details**

This function simplifies writing a kml file of site and/or sites with linked photos. Further documentation is provided in [this tutorial](#).

**Value**

A KML file of of sites with embedded associated site photos.

**Author(s)**

Jay Skovlin, D.E. Beaudette

---

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 (`KST.order = FALSE`) or ordinal (`KST.order = TRUE`) scale variables, based on soil order, suborder, greatgroup, and subgroup taxa.

**Usage**

```
SoilTaxonomyDendrogram(
  spc,
  KST.order = TRUE,
  rotationOrder = NULL,
  level = c(soilorder = "soilorder", suborder = "suborder", greatgroup = "greatgroup",
    subgroup = "subgroup"),
  cluster.method = c("divisive", "agglomerative"),
  cluster.args = list(),
  name = "hzname",
  name.style = "center-center",
  id.style = "side",
  max.depth = max(spc),
```

```

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,
shrink = FALSE,
font.id = 2,
cex.taxon.labels = 0.66,
dend.color = par("fg"),
dend.width = 1,
dend.type = c("phylogram", "cladogram"),
...
)

```

### Arguments

<code>spc</code>	a <code>SoilProfileCollection</code> object, typically returned by <code>soilDB::fetchOSD</code>
<code>KST.order</code>	logical, encode / cluster taxa via ordinal factors, based on ordering within Keys to Soil Taxonomy
<code>rotationOrder</code>	character vector of profile IDs with desired ordering of leaves in the dendrogram from left to right; exact ordering is not always possible
<code>level</code>	character. One or more site-level columns in <code>spc</code> . Default: "soilorder", "suborder", "greatgroup" and "subgroup"
<code>cluster.method</code>	Either "divisive" ( <code>cluster::diana()</code> ; default) or "agglomerative" ( <code>cluster::agnes()</code> )
<code>cluster.args</code>	Optional: additional arguments for <code>cluster::diana()</code> or <code>cluster::agnes()</code> cluster methods
<code>name</code>	column name containing horizon names
<code>name.style</code>	passed to <code>aqp::plotSPC</code>
<code>id.style</code>	passed to <code>aqp::plotSPC</code>
<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>shrink</code>	logical, should long horizon names be shrunk by 80% ?
<code>font.id</code>	font style applied to profile id, default is 2 (bold)

```

cex.taxon.labels      character scaling for taxonomic information
dend.color            dendrogram line color
dend.width            dendrogram line width
dend.type             dendrogram type, passed to plot.phylo(), either "phylogram" or "cladogram"
...                  additional arguments to aqp::plotSPC

```

### Details

This function looks for specific site-level attributes named: "soilorder", "suborder", "greatgroup", and "subgroup", or their NASIS physical column name analogues "taxorder", "taxsuborder", "taxgrtgroup", and "taxsubgrp". See <https://github.com/ncss-tech/sharpsshootR/blob/master/misc/soilTaxonomyDendrogram-examples.R> for some examples.

The rotationOrder argument uses `ape::rotateConstr()` to reorder leaves within the hclust representation of the ST hierarchy. Perfect sorting is not always possible.

### Value

An invisibly-returned list containing:

- dist: pair-wise dissimilarity matrix
- order: final ordering of hclust leaves

### Author(s)

D.E. Beaudette

### Examples

```

# built-in data, same as results from soilDB::fetchOSD()
data("OSDexamples")

# examples using first 8 profiles

# KST-style ordering
SoilTaxonomyDendrogram(
  OSDexamples$SPC[1:8, ], width = 0.3, name.style = 'center-center',
  KST.order = TRUE
)

# classic ordering, based on nominal scale variables (unordered factors)
SoilTaxonomyDendrogram(
  OSDexamples$SPC[1:8, ], width = 0.3, name.style = 'center-center',
  KST.order = FALSE
)

```



---

table5.2	<i>Table 5.2 from Hole and Campbell, 1985.</i>
----------	--

---

**Description**

An adjacency matrix describing shared soil map boundary segments from the Soil Survey of Shawnee county, KS. This is table 5.2 from Hole and Campbell, 1985.

**Usage**

```
data(table5.2)
```

**Format**

An object of class `matrix` (inherits from `array`) with 18 rows and 18 columns.

**References**

Hole, F.D. and J.B. Campbell. Soil Landscape Analysis. Rowman and Allanheld, 1985.

**Examples**

```
data("table5.2")

if(requireNamespace("igraph")) {

  # note special incantation to get the "correct" graph structure
  g <- igraph::graph_from_adjacency_matrix(table5.2, mode = 'upper', diag = FALSE, weighted = TRUE)

  # visualize
  op <- par(no.readonly = TRUE)

  par(mar = c(0,0,0,0))
  plot(g)

  plot(g, vertex.size = sqrt(igraph::degree(g) * 25), vertex.label.family = 'sans')

  # find communities
  cm <- igraph::cluster_walktrap(g)
  plot(cm, g, vertex.label.family = 'sans')

  par(op)
}
```

---

vizAnnualClimate      *Annual Climate Summaries for Soil Series Data*

---

### Description

Annual climate summaries for soil series, based on `latticeExtra::segplot`, based on 5th, 25th, 50th, 75th, and 95th percentiles. Input data should be from `soilDB::fetchOSD`.

### Usage

```
vizAnnualClimate(climate.data, IQR.cex = 1, s = NULL, s.col = "firebrick", ...)
```

### Arguments

<code>climate.data</code>	Annual climate summaries, as returned from <code>soilDB::fetchOSD(..., extended=TRUE)</code>
<code>IQR.cex</code>	scaling factor for bar representing interquartile range
<code>s</code>	a soil series name, e.g. "LUCY", to highlight
<code>s.col</code>	color for highlighted soil series
<code>...</code>	further arguments passed to <code>latticeExtra::segplot</code>

### Details

This function was designed for use with `soilDB::fetchOSD`. It might be possible to use with other sources of data but your mileage may vary. See the [Soil Series Query Functions](#) tutorial for more information.

### Value

A list with the following elements:

- `fig`: lattice object (the figure)
- `clust`: clustering object returned by `cluster::diana`

### Author(s)

D.E. Beaudette

### See Also

`vizHillslopePosition`

---

vizFlatsPosition      *Visual Summary of Flat Landform Positions*

---

### Description

A unique display of landform position probability.

### Usage

```
vizFlatsPosition(
  x,
  s = NULL,
  annotations = TRUE,
  annotation.cex = 0.75,
  cols = c("#2B83BA", "#ABDDA4", "#FFFFBF", "#FDAE61", "#D7191C"),
  ...
)
```

### Arguments

x	data.frame as created by <code>soilDB::fetchOSD(..., extended=TRUE)</code> , see details
s	an optional soil series name, highlighted in the figure
annotations	logical, add number of record and normalized Shannon entropy values
annotation.cex	annotation label scaling factor
cols	vector of colors
...	additional arguments to <code>[iterateHydOrder]</code> : <code>target = 0.9</code> , <code>maxIter = 20</code> , <code>j.amount = 0.05</code> , <code>verb</code>

### Details

See the [Soil Series Query Functions](#) tutorial for more information.

### Value

A list with the following elements:

- `fig`: lattice object (the figure)
- `order`: 1D ordering from `cluster::diana`
- `clust`: `hclust` object
- `match.rate`: fraction of series matching target hydrologic ordering, after clustering + rotation

### Author(s)

D.E. Beaudette

---

 vizGeomorphicComponent

*Visual Summary of Hill Landform Positions*


---

## Description

A unique display of landform position probability.

## Usage

```

vizGeomorphicComponent(
  x,
  s = NULL,
  annotations = TRUE,
  annotation.cex = 0.75,
  cols = c("#D53E4F", "#FC8D59", "#FEE08B", "#E6F598", "#99D594", "#3288BD"),
  ...
)

```

## Arguments

x	data.frame as created by <code>soilDB::fetchOSD(..., extended=TRUE)</code> , see details
s	an optional soil series name, highlighted in the figure
annotations	logical, add number of record and normalized Shannon entropy values
annotation.cex	annotation label scaling factor
cols	vector of colors
...	additional arguments to <code>[iterateHydOrder]</code> : <code>target = 0.9</code> , <code>maxIter = 20</code> , <code>j.amount = 0.05</code> , <code>verb</code>

## Details

See the [Soil Series Query Functions](#) tutorial for more information.

## Value

A list with the following elements:

- `fig`: lattice object (the figure)
- `order`: 1D ordering from `cluster::diana`
- `clust`: `hclust` object
- `match.rate`: fraction of series matching target hydrologic ordering, after clustering + rotation

## Author(s)

D.E. Beaudette

---

vizHillslopePosition *Visual Summary of Hillslope Position*

---

### Description

A unique display of hillslope position probability.

### Usage

```
vizHillslopePosition(
  x,
  s = NULL,
  annotations = TRUE,
  annotation.cex = 0.75,
  cols = c("#2B83BA", "#ABDDA4", "#FFFFFF", "#FDAE61", "#D7191C"),
  ...
)
```

### Arguments

x	data.frame as created by <code>soilDB::fetchOSD(..., extended = TRUE)</code>
s	an optional soil series name, highlighted in the figure
annotations	logical, add number of record and normalized Shannon entropy values
annotation.cex	annotation label scaling factor
cols	vector of colors
...	additional arguments to <code>[iterateHydOrder]</code> : <code>target = 0.9</code> , <code>maxIter = 20</code> , <code>j.amount = 0.05</code> , <code>verb</code>

### Details

See the [Soil Series Query Functions](#) tutorial for more information.

### Value

A list with the following elements:

- `fig`: lattice object (the figure)
- `order`: 1D ordering from `cluster::diana`
- `clust`: `hclust` object
- `match.rate`: fraction of series matching target hydrologic ordering, after clustering + rotation

### Author(s)

D.E. Beaudette

---

vizMountainPosition     *Visual Summary of Mountain Slope Positions*

---

### Description

A unique display of mountain slope position probability.

### Usage

```
vizMountainPosition(
  x,
  s = NULL,
  annotations = TRUE,
  annotation.cex = 0.75,
  cols = c("#D53E4F", "#FC8D59", "#FEE08B", "#E6F598", "#99D594", "#3288BD"),
  ...
)
```

### Arguments

x	data.frame as created by <code>soilDB::fetchOSD(..., extended=TRUE)</code> , see details
s	an optional soil series name, highlighted in the figure
annotations	logical, add number of record and normalized Shannon entropy values
annotation.cex	annotation label scaling factor
cols	vector of colors
...	additional arguments to <code>[iterateHydOrder]</code> : <code>target = 0.9</code> , <code>maxIter = 20</code> , <code>j.amount = 0.05</code> , <code>verb</code>

### Details

See the [Soil Series Query Functions](#) tutorial for more information.

### Value

A list with the following elements:

- `fig`: lattice object (the figure)
- `order`: 1D ordering from `cluster::diana`
- `clust`: `hclust` object
- `match.rate`: fraction of series matching target hydrologic ordering, after clustering + rotation

### Author(s)

D.E. Beaudette

---

vizSurfaceShape	<i>Visual Summary of Surface Shape</i>
-----------------	--

---

**Description**

A unique display of surface shape (typically curvature) probability, suitable for across-slope or down-slope shape. Use the `title` argument to make this clear.

**Usage**

```

vizSurfaceShape(
  x,
  title = "Surface Shape",
  s = NULL,
  annotations = TRUE,
  annotation.cex = 0.75,
  cols = c("#2B83BA", "#FFFFBF", "#D7191C", "#808080", "darkgreen"),
  ...
)

```

**Arguments**

<code>x</code>	data.frame as created by <code>soilDB::fetchOSD(..., extended=TRUE)</code> , see details
<code>title</code>	a reasonable title for the figure
<code>s</code>	an optional soil series name, highlighted in the figure
<code>annotations</code>	logical, add number of record and normalized Shannon entropy values
<code>annotation.cex</code>	annotation label scaling factor
<code>cols</code>	vector of colors
<code>...</code>	additional arguments to <code>[iterateHydOrder]</code> : <code>target = 0.9</code> , <code>maxIter = 20</code> , <code>j.amount = 0.05</code> , <code>verb</code>

**Details**

See the [Soil Series Query Functions](#) tutorial for more information.

**Value**

A list with the following elements:

- `fig`: lattice object (the figure)
- `order`: 1D ordering from `cluster::diana`
- `clust`: `hclust` object
- `match.rate`: fraction of series matching target hydrologic ordering, after clustering + rotation

**Author(s)**

D.E. Beaudette

---

vizTerracePosition      *Visual Summary of Terraced Landform Positions*

---

### Description

A unique display of terraced landform position probability.

### Usage

```
vizTerracePosition(
  x,
  s = NULL,
  annotations = TRUE,
  annotation.cex = 0.75,
  cols = c("#2B83BA", "#FDAE61"),
  ...
)
```

### Arguments

x	data.frame as created by <code>soilDB::fetchOSD(..., extended=TRUE)</code> , see details
s	an optional soil series name, highlighted in the figure
annotations	logical, add number of record and normalized Shannon entropy values
annotation.cex	annotation label scaling factor
cols	vector of colors
...	additional arguments to <code>[iterateHydOrder]</code> : <code>target = 0.9</code> , <code>maxIter = 20</code> , <code>j.amount = 0.05</code> , <code>verb</code>

### Details

See the [Soil Series Query Functions](#) tutorial for more information.

### Value

A list with the following elements:

- `fig`: lattice object (the figure)
- `order`: 1D ordering from `cluster::diana`
- `clust`: `hclust` object
- `match.rate`: fraction of series matching target hydrologic ordering, after clustering + rotation

### Author(s)

D.E. Beaudette



# Index

- \* **datasets**
  - amador, 6
  - CDEC.snow.courses, 8
  - HHM, 31
  - OSDexamples, 44
  - table5.2, 73
- \* **hplots**
  - aggregateColorPlot, 4
  - aspect.plot, 6
  - diagnosticPropertyPlot, 17
  - diagnosticPropertyPlot2, 19
  - dueling.dendrograms, 22
  - PCP\_plot, 44
  - percentileDemo, 45
  - plotProfileDendrogram, 49
  - plotSoilRelationChordGraph, 51
  - plotWB, 58
- \* **hplot**
  - plotSoilRelationGraph, 52
- \* **manip**
  - component.adj.matrix, 13
  - constantDensitySampling, 14
  - dist.along.grad, 21
  - geomorphBySoilSeries-SSURGO, 29
  - multinomial2logical, 43
  - sample.by.poly, 65
  - sampleRasterStackByMU, 66
  - site\_photos\_kml, 69
- aggregateColorPlot, 4
- amador, 6
- aqp::mixMunsell(), 12
- aspect.plot, 6
- CDEC.snow.courses, 8
- CDEC\_StationInfo, 9, 10, 11
- CDECquery, 9
- CDECsnowQuery, 10, 10
- colorMixtureVenn, 12
- component.adj.matrix, 13
- constantDensitySampling, 14, 65–67
- contour(), 32
- dailyWB, 15
- dailyWB(), 38, 39
- dailyWB\_SSURGO, 16
- dailyWB\_SSURGO(), 38, 39
- diagnosticPropertyPlot, 17, 43
- diagnosticPropertyPlot2, 19
- digest::digest(), 28
- dist.along.grad, 21
- dueling.dendrograms, 22
- ESS\_by\_Moran\_I, 23
- estimateSoilMoistureState, 15, 16, 23, 39
- FFD, 25, 26
- FFDplot, 26
- formatPLSS, 27, 38, 62
- generateLineHash, 28
- geomorphBySoilSeries-SSURGO, 29
- geomPosHillProbability
  - (geomorphBySoilSeries-SSURGO), 29
- geomPosMountainProbability
  - (geomorphBySoilSeries-SSURGO), 29
- HenryTimeLine, 30
- HHM, 31
- hillslope.probability
  - (geomorphBySoilSeries-SSURGO), 29
- hillslopeProbability
  - (geomorphBySoilSeries-SSURGO), 29
- huePositionPlot, 32
- hydOrder, 33
- hydOrder(), 49

- isMineralSoilMaterial, [34](#)
- iterateHydOrder, [35](#)
- iterateHydOrder(), [34](#), [36](#), [49](#)
  
- joinAdjacency, [36](#)
  
- LL2PLSS, [37](#), [62](#)
  
- moistureStateProportions, [38](#)
- moistureStateProportions(), [39](#)
- moistureStateStats, [39](#)
- moistureStateThreshold, [39](#)
- monthlyWB, [40](#)
- monthlyWB(), [61](#)
- monthlyWB\_summary (monthlyWB), [40](#)
- Moran\_I\_ByRaster, [42](#)
- multinomial2logical, [18](#), [20](#), [43](#)
  
- OSDexamples, [44](#)
  
- PCP\_plot, [44](#)
- percentileDemo, [45](#)
- plotAvailWater, [46](#)
- plotGeomorphCrossSection, [49](#)
- plotProfileDendrogram, [49](#)
- plotSoilRelationChordGraph, [51](#)
- plotSoilRelationGraph, [52](#)
- plotSoilRelationGraph(), [37](#), [63](#)
- plotTransect, [21](#), [22](#), [55](#)
- plotWB, [58](#)
- plotWB\_lines, [60](#)
- PLSS2LL, [27](#), [38](#), [62](#)
- plssMeridians (LL2PLSS), [37](#)
- polygonAdjacency, [62](#)
- prepare\_SSURGO\_hydro\_data, [64](#)
- prepareDailyClimateData, [63](#)
  
- reconcileOSDGeomorph, [64](#)
  
- sample.by.poly, [15](#), [65](#), [67](#)
- sampleRasterStackByMU, [66](#)
- samplingStability, [67](#)
- sharpshootR (sharpshootR-package), [3](#)
- sharpshootR-package, [3](#)
- simpleWB, [68](#)
- site\_photos\_kml, [69](#)
- SoilTaxonomyDendrogram, [70](#)
- spdep::poly2nb(), [63](#)
  
- surfaceShapeProbability  
(geomorphBySoilSeries-SSURGO),  
[29](#)
  
- table5.2, [73](#)
  
- vizAnnualClimate, [74](#)
- vizFlatsPosition, [75](#)
- vizGeomorphicComponent, [76](#)
- vizHillslopePosition, [77](#)
- vizMountainPosition, [78](#)
- vizSurfaceShape, [79](#)
- vizTerracePosition, [80](#)
  
- waterDayYear, [45](#)