Package 'soiltexture'

February 16, 2024

Version 1.5.3

Date 2024-02-16

Title Functions for Soil Texture Plot, Classification and Transformation

Author Julien Moeys [aut, cre],
Wei Shangguan [ctb],
Rainer Petzold [ctb],
Budiman Minasny [ctb],
Bogdan Rosca [ctb],
Nic Jelinski [ctb],
Wiktor Zelazny [ctb],
Rodolfo Marcondes Silva Souza [ctb],
Jose Lucas Safanelli [ctb],
Alexandre ten Caten [ctb]

Maintainer Julien Moeys <jules_m78-soiltexture@yahoo.fr>

Depends R (>= 4.3.0)

Suggests xtable

Description ``The Soil Texture Wizard'' is a set of R functions designed to produce texture triangles (also called texture plots, texture diagrams, texture ternary plots), classify and transform soil textures data. These functions virtually allows to plot any soil texture triangle (classification) into any triangle geometry (isosceles, right-angled triangles, etc.). This set of function is expected to be useful to people using soil textures data from different soil texture classification or different particle size systems. Many (> 15) texture triangles from all around the world are predefined in the package. A simple text based graphical user interface is provided: soiltexture_gui().

License AGPL (>= 3)

Encoding UTF-8

URL https://github.com/julienmoeys/soiltexture

BugReports https://github.com/julienmoeys/soiltexture/issues

Imports sp, MASS, tools, tcltk, utils

RoxygenNote 7.3.1

NeedsCompilation no

R **topics documented:**

---

soiltextureInfo	*Display and / or export system and package version information*

Description

Display and / or export system and package version information. Can be used to provide an overview of the system and the R packages that were used to produce some calculations, thus improving the traceability of that work in the long run.

Usage

```
soiltextureInfo(
  file = NULL,
  verbose = TRUE,
  depends = FALSE,
  md5 = TRUE,
  packages = "soiltexture"
)
```

Arguments

file	Single character string. Name of the text file (with or without its path) in which the information will be exported. If NULL (default), information are not exported.
verbose	Single logical value. If TRUE, information are displayed on the screen.
depends	Single logical value. If TRUE, information on packages dependencies are also displayed, in the same way
md5	Single logical value. If TRUE, the package MD5 checksums are returned too
packages	Single character string. Name of the package whose information must be returned.

Value

Invisibly returns the information as a vector of character strings

Author(s)

Julien Moeys [aut, cre], Wei Shangguan [ctb], Rainer Petzold [ctb], Budiman Minasny [ctb], Bogdan Rosca [ctb], Nic Jelinski [ctb], Wiktor Zelazny [ctb], Rodolfo Marcondes Silva Souza [ctb], Jose Lucas Safanelli [ctb], Alexandre ten Caten [ctb]

See Also

The base functions that were used internally to compile the information: Sys.time, Sys.info, version, zpackages, installed.packages, package_dependencies. See also the MD5 file in each package directory (and md5sum for generating these MD5 checksums).

Examples

```
library( "soiltexture" )

#   Temporary file where the info will be exported:
f <- tempfile()

#   Generate package information
soiltextureInfo( file = f, depends = TRUE, verbose = FALSE )

#   Read again the info (as for verbose = TRUE)
cat( readLines( f ), sep = "\n" )
unlink( f )

#   Also works with other packages
soiltextureInfo( packages = "sp" )
```

---

soiltexture_gui	*Text-based menu for plotting and classifying soil texture data*

Description

Text-based menu for plotting and classifying soil texture data.

Usage

```
soiltexture_gui(main = NULL, graphics = FALSE, ...)
```

Arguments

main	Single character string. Main title of the texture diagram. Set to NA to obtain a a slightly bigger figure, with no title. See TT.plot.
graphics	See select.list.
...	Additional parameters passed to soiltexture:::.read.table.menu or (subsequently) to read.table.

Details

If you simply want to obtain a figure with an empty soil texture triangle, just call soiltexture_gui() and follow the instructions.

If you want to a figure with your own soil texture data on top of a texture triangle, you must first prepare a tabular text file containing your texture data, as .txt or .csv. Such a file can be prepared with MS Excel or Libre Office, and exported as CSV ("CSV (comma delimited) (*.csv)" or "CSV (MS-DOS) (*.csv)" for example). The table **must** contain headers (column names) and it **must** the following columns and headers: CLAY, SILT and SAND. Other columns are allowed and will be ignored. In the texture data file, each row represent a record (a sample) and each column a variable.

You will be asked about the format of this text file, in particular about the field / column separator (it can be commas, semi-colons, tabulations or (multiple) spaces) and the decimal mark (comma or dot). The file encoding can be either the native encoding of the computer, or UTF-8 (without BOM).

The sum of the texture of each row must be either 1 (if expressed as a fraction) or 100 (if expressed as a percentage). You will be asked about the unit. Only small divergences from 1 or 100 are allowed, but you will be asked if you want to normalise your data beforehand, so larger divergences are possible.

You will also be asked which texture classification system you want to use (FAO, USDA, etc.). It is possible to plot a texture triangle without texture classification.

Finally, if you have chosen a texture classification system, `soiltexture_gui` can classify each record according to this classification system and **return you the texture class of each record**, as a CSV text file.

The texture triangle is show to you with R default graphical device, and you can choose to export a PNG figure of the resulting texture triangle (between 512 and 2048 pixel width/height, depending on what you chose).

Value

Either NULL if no texture data was imported, or a [data.frame](#) (if texture data was imported). The texture classification is also returned (when the user asked for a texture classification).

Author(s)

Julien Moeys [aut, cre], Wei Shangguan [ctb], Rainer Petzold [ctb], Budiman Minasny [ctb], Bogdan Rosca [ctb], Nic Jelinski [ctb], Wiktor Zelazny [ctb], Rodolfo Marcondes Silva Souza [ctb], Jose Lucas Safanelli [ctb], Alexandre ten Caten [ctb]

Examples

```
library( "soiltexture" )

#   Call the text graphical user interface

soiltexture_gui()

#   ... and follow the instructions indicated to you!
```

`TT.add`	*Function to add a new default package parameters.*

Description

Function to add a new default package parameters. Mostly used to add a new texture triangle definition.

Usage

```
TT.add(..., par.list = "TT.par", bkp.par.list = "TT.par.bkp", par.env = TT.env)
```

Arguments

`...`	parameters to be changed in format: "parameter_name1" = new_value1, "parameter_name2" = new_value2
`par.list`	Name of the list containing the parameters
`bkp.par.list`	Name of the backuped list containing the default parameters
`par.env`	name of the environment containing the parameter lists

Author(s)

Julien Moeys [aut, cre], Wei Shangguan [ctb], Rainer Petzold [ctb], Budiman Minasny [ctb], Bogdan Rosca [ctb], Nic Jelinski [ctb], Wiktor Zelazny [ctb], Rodolfo Marcondes Silva Souza [ctb], Jose Lucas Safanelli [ctb], Alexandre ten Caten [ctb]

`TT.baseplot`	*Internal. Create an empty plot scene for a texture triangle.*

Description

Create an empty plot where a texture triangle can be drawn with other secondary functions (frame, axis, ...). Also return the 'geo' parameters needed by these secondary functions.

Usage

```
TT.baseplot(
  geo = NULL,
  class.sys = "none",
  blr.clock = NULL,
  tlr.an = NULL,
  blr.tx = NULL,
  text.sum = NULL,
  base.css.ps.lim = NULL,
  tri.sum.tst = NULL,
  tri.pos.tst = NULL,
  text.tol = NULL,
  unit.ps = NULL,
  unit.tx = NULL,
  b.lim = NULL,
  l.lim = NULL,
  main = NULL,
  new.mar = NULL,
  bg = NULL,
  fg = NULL,
```

```
    col = NULL,
    cex.main = NULL,
    lang = NULL
)
```

Arguments

`geo`	See `TT.plot`
`class.sys`	See `TT.plot`
`blr.clock`	See `TT.plot`
`tlr.an`	See `TT.plot`
`blr.tx`	See `TT.plot`
`text.sum`	See `TT.plot`
`base.css.ps.lim`	
	See `TT.plot`
`tri.sum.tst`	See `TT.plot`
`tri.pos.tst`	See `TT.plot`
`text.tol`	See `TT.plot`
`unit.ps`	See `TT.plot`
`unit.tx`	See `TT.plot`
`b.lim`	See `TT.plot`
`l.lim`	See `TT.plot`
`main`	See `TT.plot`
`new.mar`	See `TT.plot`
`bg`	See `TT.plot`
`fg`	See `TT.plot`
`col`	See `TT.plot`
`cex.main`	See `TT.plot`
`lang`	See `TT.plot`

Author(s)

Julien Moeys [aut, cre], Wei Shangguan [ctb], Rainer Petzold [ctb], Budiman Minasny [ctb], Bogdan Rosca [ctb], Nic Jelinski [ctb], Wiktor Zelazny [ctb], Rodolfo Marcondes Silva Souza [ctb], Jose Lucas Safanelli [ctb], Alexandre ten Caten [ctb]

TT.chemometrics.alr   *Compute the additive log-ratio transformation of compositional data.*

Description

Function that compute the additive log-ratio transformation of compositional data (here texture data). This a a copy-paste-and-rename of the alr function provided by the package chemometrics: P. Filzmoser and K. Varmuza (2008). chemometrics: Multivariate Statistical Analysis in Chemometrics. R package version 0.4. The function has been modified so it returns NA when a value is below or equal to zero (this happens when using a regular grid of texture data, for practical reasons). The function has also been modified so it uses column name rather than column index.

Usage

```
TT.chemometrics.alr(X, divisorvar, css.names)
```

Arguments

X	A data.frame containing soil texture data
divisorvar	number of ratioing variable
css.names	See `TT.plot`

Author(s)

Julien Moeys [aut, cre], Wei Shangguan [ctb], Rainer Petzold [ctb], Budiman Minasny [ctb], Bogdan Rosca [ctb], Nic Jelinski [ctb], Wiktor Zelazny [ctb], Rodolfo Marcondes Silva Souza [ctb], Jose Lucas Safanelli [ctb], Alexandre ten Caten [ctb]

TT.classes   *Plot the texture classes polygons in a texture triangle plot.*

Description

Plot the texture classes ploygons in an existing texture triangle plot. Draw the polygons and the labels inside each polygons.

Usage

```
TT.classes(
  geo,
  class.sys,
  tri.css.ps.lim = NULL,
  css.transf = NULL,
  text.transf.fun = NULL,
  trsf.add.opt1 = NULL,
```

```
        trsf.add.opt2 = NULL,
        text.tol = NULL,
        text.sum = NULL,
        base.css.ps.lim = NULL,
        blr.tx = NULL,
        blr.clock = NULL,
        tri.sum.tst = NULL,
        tri.pos.tst = NULL,
        bg = NULL,
        class.lab.col = NULL,
        class.p.bg.col = NULL,
        class.p.bg.hue = NULL,
        class.line.col = NULL,
        class.lty = NULL,
        class.lab.show = NULL,
        cex.lab = NULL,
        font.lab = NULL,
        family.op = NULL,
        lwd.axis = NULL,
        col.axis = NULL,
        new.centroid = TRUE
    )
```

Arguments

geo	See TT.plot
class.sys	See TT.plot
tri.css.ps.lim	See TT.plot
css.transf	See TT.plot
text.transf.fun	
	See TT.plot
trsf.add.opt1	See TT.plot
trsf.add.opt2	See TT.plot
text.tol	See TT.plot
text.sum	See TT.plot
base.css.ps.lim	
	See TT.plot
blr.tx	See See TT.plot
blr.clock	See See TT.plot
tri.sum.tst	See TT.plot
tri.pos.tst	See TT.plot
bg	See TT.plot
class.lab.col	See TT.plot
class.p.bg.col	See TT.plot

```
class.p.bg.hue  See TT.plot
class.line.col  See TT.plot
class.lty       See TT.plot
class.lab.show  See TT.plot
cex.lab         See TT.plot
font.lab        See TT.plot
family.op       See TT.plot
lwd.axis        See TT.plot
col.axis        See TT.plot
```

new.centroid   Single logical. If TRUE (default) the new method (Paul Bourke) is used to calculate the centroid. If FALSE the centroid is taken as the mean x and y coordinates of the vertices.

Author(s)

Julien Moeys [aut, cre], Wei Shangguan [ctb], Rainer Petzold [ctb], Budiman Minasny [ctb], Bogdan Rosca [ctb], Nic Jelinski [ctb], Wiktor Zelazny [ctb], Rodolfo Marcondes Silva Souza [ctb], Jose Lucas Safanelli [ctb], Alexandre ten Caten [ctb]

TT.classes.tbl   *Returns the table of classes of a texture classification system.*

Description

Returns the table of classes of a texture classification system. Returns the classes abbreviations, names and the vertices numbers that defines each class. Use TT.vertices.tbl() to retrieve the clay silt sand coordinates of the triangle classes vertices. See also TT.vertices.plot().

Usage

```
TT.classes.tbl(class.sys = "HYPRES.TT", collapse = NULL)
```

Arguments

class.sys   See TT.plot

collapse   Single character string. Default ", ". Separator when showing differents points in each class.

Author(s)

Julien Moeys [aut, cre], Wei Shangguan [ctb], Rainer Petzold [ctb], Budiman Minasny [ctb], Bogdan Rosca [ctb], Nic Jelinski [ctb], Wiktor Zelazny [ctb], Rodolfo Marcondes Silva Souza [ctb], Jose Lucas Safanelli [ctb], Alexandre ten Caten [ctb]

TT.contour   *Wrapper for the contour() function adapted to texture triangles.*

Description

A wrapper for the contour() function adapted to texture triangles (plot preparation). designed to plot the results of TT.mahalanobis() or TT.kde2d(), before or after plot.

Usage

```
TT.contour(
  geo,
  x,
  add = FALSE,
  tri.sum.tst = NULL,
  tri.pos.tst = NULL,
  text.tol = NULL,
  unit.ps = NULL,
  unit.tx = NULL,
  b.lim = NULL,
  l.lim = NULL,
  main = NULL,
  new.mar = NULL,
  bg = NULL,
  fg = NULL,
  col = NULL,
  cex.main = NULL,
  lang = NULL,
  nlevels = 10,
  levels = NA,
  labels = NULL,
  xlim = NA,
  ylim = NA,
  zlim = NA,
  labcex = 1,
  drawlabels = TRUE,
  method = "flattest",
  axes = TRUE,
  frame.plot = NA,
  lty = NA,
  lwd = NA,
  blr.clock = NULL,
  tlr.an = NULL,
  blr.tx = NULL,
  text.sum = NULL,
  base.css.ps.lim = NULL,
  ...
```

```
)
```

Arguments

`geo`	See `TT.plot`
`x`	A data.frame of matrix with 3 columns: x, y and z.
`add`	If TRUE adds the contour to an existing plot.
`tri.sum.tst`	See `TT.plot`
`tri.pos.tst`	See `TT.plot`
`text.tol`	See `TT.plot`
`unit.ps`	See `TT.plot`
`unit.tx`	See `TT.plot`
`b.lim`	See `TT.plot`
`l.lim`	See `TT.plot`
`main`	See `TT.plot`
`new.mar`	See `TT.plot`
`bg`	See `TT.plot`
`fg`	See `TT.plot`
`col`	See `TT.plot`
`cex.main`	See `TT.plot`
`lang`	See `TT.plot`
`nlevels`	See `contour`
`levels`	See `contour`
`labels`	See `contour`
`xlim`	See `contour`
`ylim`	See `contour`
`zlim`	See `contour`
`labcex`	See `contour`
`drawlabels`	See `contour`
`method`	See `contour`
`axes`	See `contour`
`frame.plot`	See `contour`
`lty`	See `contour`
`lwd`	See `contour`
`blr.clock`	See `TT.plot`
`tlr.an`	See `TT.plot`
`blr.tx`	See `TT.plot`
`text.sum`	See `TT.plot`
`base.css.ps.lim`	See `TT.plot`
`...`	Additional parameters passed to See `contour`

Author(s)

Julien Moeys [aut, cre], Wei Shangguan [ctb], Rainer Petzold [ctb], Budiman Minasny [ctb], Bogdan Rosca [ctb], Nic Jelinski [ctb], Wiktor Zelazny [ctb], Rodolfo Marcondes Silva Souza [ctb], Jose Lucas Safanelli [ctb], Alexandre ten Caten [ctb]

TT.data.test	*Test the validity of some soil texture data table (3 particle size classes).*

Description

Test the validity of some soil texture data table. (1) Test that it is a data.frame or matrix, (2) Test that column names contains 'css.names', (3) Test that there are no missing values, (4) that all values are >= 0, (5) That the sum of the 3 particle size classes is >= 'text.sum'*(1-'text.tol') or <= 'text.sum'*(1+'text.tol'). 'tri.data' may contain other variables than the 3 textuer classes (ignored).

Usage

```
TT.data.test(
  tri.data,
  css.names = NULL,
  text.sum = NULL,
  text.tol = NULL,
  tri.sum.tst = NULL,
  tri.pos.tst = NULL
)
```

Arguments

`tri.data`	See `TT.plot`
`css.names`	See `TT.plot`
`text.sum`	See `TT.plot`
`text.tol`	See `TT.plot`
`tri.sum.tst`	See `TT.plot`
`tri.pos.tst`	See `TT.plot`

Author(s)

Julien Moeys [aut, cre], Wei Shangguan [ctb], Rainer Petzold [ctb], Budiman Minasny [ctb], Bogdan Rosca [ctb], Nic Jelinski [ctb], Wiktor Zelazny [ctb], Rodolfo Marcondes Silva Souza [ctb], Jose Lucas Safanelli [ctb], Alexandre ten Caten [ctb]

TT.data.test.X   *Test the validity of some soil texture data table (X particle size classes).*

Description

Test the validity of some soil texture data table. (1) Test that it is a data.frame or matrix, (3) Test that there are no missing values, (4) that all values are >= 0, (5) That the sum of the X particle size class is >= 'text.sum'*(1-'text.tol') or <= 'text.sum'*(1+'text.tol'). Contrary to TT.data.test() no test are performed for the particle size classes and columns names, so 'tri.data' should only contains texture data, and nothing else.

Usage

```
TT.data.test.X(
  tri.data,
  text.sum = NULL,
  text.tol = NULL,
  tri.sum.tst = NULL,
  tri.pos.tst = NULL
)
```

Arguments

tri.data	See TT.plot
text.sum	See TT.plot
text.tol	See TT.plot
tri.sum.tst	See TT.plot
tri.pos.tst	See TT.plot

Author(s)

Julien Moeys [aut, cre], Wei Shangguan [ctb], Rainer Petzold [ctb], Budiman Minasny [ctb], Bogdan Rosca [ctb], Nic Jelinski [ctb], Wiktor Zelazny [ctb], Rodolfo Marcondes Silva Souza [ctb], Jose Lucas Safanelli [ctb], Alexandre ten Caten [ctb]

TT.dataset   *Genetates a virtual cross correlated clay silt sand + Z dataset.*

Description

Genetates a virtual cross correlated clay silt sand + Z dataset, where Z is a virtual 4th variable correlated to the texture.

Usage

```
TT.dataset(n, seed.val = NULL, css.names = NULL, text.sum = NULL)
```

Arguments

n	Single numeric value. Number of points to be generated.
seed.val	See set.seed
css.names	See TT.plot
text.sum	See TT.plot

Author(s)

Julien Moeys [aut, cre], Wei Shangguan [ctb], Rainer Petzold [ctb], Budiman Minasny [ctb], Bogdan Rosca [ctb], Nic Jelinski [ctb], Wiktor Zelazny [ctb], Rodolfo Marcondes Silva Souza [ctb], Jose Lucas Safanelli [ctb], Alexandre ten Caten [ctb]

TT.dia2phi	*Convert a soil particle diameter dia [micro-meters] into phi = -log2(dia/1000)*

Description

Convert a soil particle diameter dia [micro-meters] into phi = -log2(dia). See also TT.phi2dia().

Usage

```
TT.dia2phi(dia)
```

Arguments

dia	Particle size diameter in micro-meters (will be converted in milli-meters)

Author(s)

Julien Moeys [aut, cre], Wei Shangguan [ctb], Rainer Petzold [ctb], Budiman Minasny [ctb], Bogdan Rosca [ctb], Nic Jelinski [ctb], Wiktor Zelazny [ctb], Rodolfo Marcondes Silva Souza [ctb], Jose Lucas Safanelli [ctb], Alexandre ten Caten [ctb]

TT.geo.get   *Retrieve and return the geometrical parameters from a list of parame-*
                        ter values (NULL or not).

Description

Retrieve and return the geometrical parameters from a list of parameter values (NULL or not).

Usage

```
TT.geo.get(
  class.sys = NULL,
  blr.clock = NULL,
  tlr.an = NULL,
  blr.tx = NULL,
  text.sum = NULL,
  base.css.ps.lim = NULL
)
```

Arguments

class.sys	See TT.plot
blr.clock	See TT.plot
tlr.an	See TT.plot
blr.tx	See TT.plot
text.sum	See TT.plot
base.css.ps.lim	
	See TT.plot

Author(s)

Julien Moeys [aut, cre], Wei Shangguan [ctb], Rainer Petzold [ctb], Budiman Minasny [ctb], Bogdan Rosca [ctb], Nic Jelinski [ctb], Wiktor Zelazny [ctb], Rodolfo Marcondes Silva Souza [ctb], Jose Lucas Safanelli [ctb], Alexandre ten Caten [ctb]

TT.get   *Function to retrieve / get the default package parameters.*

Description

Function to retrieve / get the default package parameters.

Usage

```
TT.get(..., par.list = "TT.par", bkp.par.list = "TT.par.bkp", par.env = TT.env)
```

Arguments

`...`	Vector of character strings. Name of arguments for which default value is to be retrieved.
`par.list`	Name of the list containing the parameters
`bkp.par.list`	Name of the backuped list containing the default parameters
`par.env`	name of the environment containing the parameter lists

Author(s)

Julien Moeys [aut, cre], Wei Shangguan [ctb], Rainer Petzold [ctb], Budiman Minasny [ctb], Bogdan Rosca [ctb], Nic Jelinski [ctb], Wiktor Zelazny [ctb], Rodolfo Marcondes Silva Souza [ctb], Jose Lucas Safanelli [ctb], Alexandre ten Caten [ctb]

`TT.grid`	*Plot a grid at regular texture intervals inside an existing soil texture triangle.*

Description

Plot a grid at regular texture intervals inside an existing soil texture triangle.

Usage

```
TT.grid(
  geo = geo,
  at = NULL,
  text.tol = NULL,
  text.sum = NULL,
  blr.clock = NULL,
  grid.col = NULL,
  grid.lty = NULL,
  lwd.axis = NULL,
  tri.sum.tst = NULL,
  tri.pos.tst = NULL,
  class.p.bg.col = NULL,
  class.p.bg.hue = NULL,
  frame.bg.col = NULL,
  bg = NULL,
  col.axis = NULL
)
```

Arguments

`geo`	See `TT.plot`
`at`	Vector of numeric values.

`text.tol`	See [TT.plot](#)
`text.sum`	See [TT.plot](#)
`blr.clock`	See [TT.plot](#)
`grid.col`	Passed to argument 'col' of [segments](#)
`grid.lty`	Passed to argument 'lty' of [segments](#)
`lwd.axis`	See [TT.plot](#)
`tri.sum.tst`	See [TT.plot](#)
`tri.pos.tst`	See [TT.plot](#)
`class.p.bg.col`	See [TT.plot](#)
`class.p.bg.hue`	See [TT.plot](#)
`frame.bg.col`	See [TT.plot](#)
`bg`	See [TT.plot](#)
`col.axis`	See [TT.plot](#)

Author(s)

Julien Moeys [aut, cre], Wei Shangguan [ctb], Rainer Petzold [ctb], Budiman Minasny [ctb], Bogdan Rosca [ctb], Nic Jelinski [ctb], Wiktor Zelazny [ctb], Rodolfo Marcondes Silva Souza [ctb], Jose Lucas Safanelli [ctb], Alexandre ten Caten [ctb]

`TT.image`	*Wrapper for the image() function adapted to texture triangles.*

Description

A wrapper for the image() function adapted to texture triangles (plot preparation). designed to plot the results of TT.mahalanobis() or TT.kde2d() [to be written], before or after plot.

Usage

```
TT.image(
  geo,
  x,
  add = FALSE,
  tri.sum.tst = NULL,
  tri.pos.tst = NULL,
  text.tol = NULL,
  unit.ps = NULL,
  unit.tx = NULL,
  b.lim = NULL,
  l.lim = NULL,
  main = NULL,
  new.mar = NULL,
```

```
    bg = NULL,
    fg = NULL,
    cex.main = NULL,
    lang = NULL,
    xlim = NA,
    ylim = NA,
    zlim = NA,
    col = rev(heat.colors(12)),
    oldstyle = FALSE,
    blr.clock = NULL,
    tlr.an = NULL,
    blr.tx = NULL,
    text.sum = NULL,
    base.css.ps.lim = NULL,
    ...
)
```

Arguments

geo	See TT.plot
x	A list with three items, "x", "y" and "z"
add	If TRUE the image is added to an existing plot.
tri.sum.tst	See TT.plot
tri.pos.tst	See TT.plot
text.tol	See TT.plot
unit.ps	See TT.plot
unit.tx	See TT.plot
b.lim	See TT.plot
l.lim	See TT.plot
main	See TT.plot
new.mar	See TT.plot
bg	See TT.plot
fg	See TT.plot
cex.main	See TT.plot
lang	See TT.plot
xlim	Passed to image
ylim	Passed to image
zlim	Passed to image
col	Passed to image
oldstyle	Passed to image
blr.clock	See TT.plot
tlr.an	See TT.plot

```
blr.tx          See TT.plot

text.sum        See TT.plot

base.css.ps.lim
                See TT.plot

...             Additional parameters passed to image().
```

Author(s)

Julien Moeys [aut, cre], Wei Shangguan [ctb], Rainer Petzold [ctb], Budiman Minasny [ctb], Bog-
dan Rosca [ctb], Nic Jelinski [ctb], Wiktor Zelazny [ctb], Rodolfo Marcondes Silva Souza [ctb],
Jose Lucas Safanelli [ctb], Alexandre ten Caten [ctb]

TT.iwd	*Inverse weighted distance interpolation on a grid.*

Description

Inverse weighted distance interpolation on a grid.

Usage

```
TT.iwd(
  tri.data,
  z.name,
  geo,
  css.names = NULL,
  text.tol = NULL,
  text.sum = NULL,
  blr.clock = NULL,
  tri.sum.tst = NULL,
  tri.pos.tst = NULL,
  set.par = FALSE,
  n = 25,
  lims = c("points", "triangle")[1],
  max.dist = NULL,
  q.max.dist = 0.5,
  pow = 0.5
)
```

Arguments

```
tri.data        See TT.plot

z.name          See TT.plot

geo             See TT.plot

css.names       See TT.plot
```

`text.tol`	See [TT.plot](#)
`text.sum`	See [TT.plot](#)
`blr.clock`	See [TT.plot](#)
`tri.sum.tst`	See [TT.plot](#)
`tri.pos.tst`	See [TT.plot](#)
`set.par`	If TRUE parameters are set automatically to their defualt value
`n`	Number of points in the grid
`lims`	"points" or "triangle"
`max.dist`	Maximum distance
`q.max.dist`	Passed to [quantile](#)
`pow`	Single numeric value. Power of the inverse weighted distance

Author(s)

Julien Moeys [aut, cre], Wei Shangguan [ctb], Rainer Petzold [ctb], Budiman Minasny [ctb], Bogdan Rosca [ctb], Nic Jelinski [ctb], Wiktor Zelazny [ctb], Rodolfo Marcondes Silva Souza [ctb], Jose Lucas Safanelli [ctb], Alexandre ten Caten [ctb]

`TT.kde2d`	*Calculated the 2D probabilty density on an x-y grid.*

Description

Function that calculated the 2D probabilty density on an x-y grid (and NOT on the clay silt sand reference system). Wrapper around the kde2d function from the MASS package.

Usage

```
TT.kde2d(
  geo,
  tri.data,
  css.names = NULL,
  text.tol = NULL,
  text.sum = NULL,
  blr.clock = NULL,
  tri.sum.tst = NULL,
  tri.pos.tst = NULL,
  set.par = FALSE,
  n = 25,
  lims = c("points", "triangle")[2]
)
```

Arguments

`geo`	See `TT.plot`
`tri.data`	See `TT.plot`
`css.names`	See `TT.plot`
`text.tol`	See `TT.plot`
`text.sum`	See `TT.plot`
`blr.clock`	See `TT.plot`
`tri.sum.tst`	See `TT.plot`
`tri.pos.tst`	See `TT.plot`
`set.par`	If TRUE parameters are set automatically to their defualt value
`n`	Passed to `kde2d`
`lims`	Passed to `kde2d`

Author(s)

Julien Moeys [aut, cre], Wei Shangguan [ctb], Rainer Petzold [ctb], Budiman Minasny [ctb], Bogdan Rosca [ctb], Nic Jelinski [ctb], Wiktor Zelazny [ctb], Rodolfo Marcondes Silva Souza [ctb], Jose Lucas Safanelli [ctb], Alexandre ten Caten [ctb]

`TT.locator`	*Interactive (mouse clic) retrieval the CLAY SILT SAND coordinate of points on a texture triangle.*

Description

Interactive (mouse clic) retrieval the CLAY SILT SAND coordinate of points on a texture triangle.

Usage

```
TT.locator(
  geo,
  css.names = NULL,
  text.tol = NULL,
  tri.sum.tst = NULL,
  tri.pos.tst = FALSE,
  set.par = FALSE,
  n = 512,
  type = "n",
  ...
)
```

Arguments

`geo`	See `TT.plot`
`css.names`	See `TT.plot`
`text.tol`	See `TT.plot`
`tri.sum.tst`	See `TT.plot`
`tri.pos.tst`	See `TT.plot`
`set.par`	If TRUE parameters are set automatically to their defualt value
`n`	Passed to `locator`
`type`	Passed to `locator`
`...`	Further argumets passed to locator()

Author(s)

Julien Moeys [aut, cre], Wei Shangguan [ctb], Rainer Petzold [ctb], Budiman Minasny [ctb], Bogdan Rosca [ctb], Nic Jelinski [ctb], Wiktor Zelazny [ctb], Rodolfo Marcondes Silva Souza [ctb], Jose Lucas Safanelli [ctb], Alexandre ten Caten [ctb]

TT.mahalanobis	*Calculates the Mahalanobis distance between clay silt and sand.*

Description

Function that calculated the Mahalanobis distance between clay silt and sand, on a regular x-y grid (back-transformed to Clay silt and sand for Mahalanobis calculation). The underlying function is mahalanobis() by R Development Core Team (2009)

Usage

```
TT.mahalanobis(
  geo,
  tri.data,
  css.names = NULL,
  text.tol = NULL,
  text.sum = NULL,
  blr.clock = NULL,
  tri.sum.tst = NULL,
  tri.pos.tst = NULL,
  set.par = FALSE,
  n = 25,
  center = NULL,
  cov.mat = NULL,
  inverted = FALSE,
  ...,
  alr = FALSE,
  divisorvar = 2
)
```

Arguments

`geo`	See `TT.plot`
`tri.data`	See `TT.plot`
`css.names`	See `TT.plot`
`text.tol`	See `TT.plot`
`text.sum`	See `TT.plot`
`blr.clock`	See `TT.plot`
`tri.sum.tst`	See `TT.plot`
`tri.pos.tst`	See `TT.plot`
`set.par`	If TRUE parameters are set automatically to their defualt value
`n`	Number of points in the grid.
`center`	Passed to `mahalanobis`
`cov.mat`	Passed to `mahalanobis`
`inverted`	Passed to `mahalanobis`
`...`	More argument passed to `mahalanobis`
`alr`	If TRUE an additive log-ratio transformation of the data is performed, and the Mahalanobis distance is computed on all classes but css.names[divisorvar]
`divisorvar`	The Mahalanobis distance will be computed on all the texture class but css.names[divisorvar]

Author(s)

Julien Moeys [aut, cre], Wei Shangguan [ctb], Rainer Petzold [ctb], Budiman Minasny [ctb], Bog-dan Rosca [ctb], Nic Jelinski [ctb], Wiktor Zelazny [ctb], Rodolfo Marcondes Silva Souza [ctb], Jose Lucas Safanelli [ctb], Alexandre ten Caten [ctb]

`TT.normalise.sum`   *Normalises the sum of the 3 particle size classes.*

Description

Normalises the sum of the 3 particle size classes in tri.data to text.sum (100%).

Usage

```
TT.normalise.sum(
  tri.data,
  css.names = NULL,
  text.sum = NULL,
  text.tol = NULL,
  tri.pos.tst = NULL,
  residuals = FALSE
)
```

Arguments

`tri.data`	See `TT.plot`
`css.names`	See `TT.plot`
`text.sum`	See `TT.plot`
`text.tol`	See `TT.plot`
`tri.pos.tst`	See `TT.plot`
`residuals`	If TRUE the residuals are calculated and added to the returned data.frame

Author(s)

Julien Moeys [aut, cre], Wei Shangguan [ctb], Rainer Petzold [ctb], Budiman Minasny [ctb], Bogdan Rosca [ctb], Nic Jelinski [ctb], Wiktor Zelazny [ctb], Rodolfo Marcondes Silva Souza [ctb], Jose Lucas Safanelli [ctb], Alexandre ten Caten [ctb]

`TT.normalise.sum.X`	*Normalises the sum of the X particle size classes.*

Description

Normalises the sum of the X particle size classes in tri.data to text.sum (100%).

Usage

```
TT.normalise.sum.X(
  tri.data,
  text.sum = NULL,
  text.tol = NULL,
  tri.pos.tst = NULL,
  residuals = FALSE
)
```

Arguments

`tri.data`	See `TT.plot`
`text.sum`	See `TT.plot`
`text.tol`	See `TT.plot`
`tri.pos.tst`	See `TT.plot`
`residuals`	If TRUE the residuals are calculated and added to the returned data.frame

Author(s)

Julien Moeys [aut, cre], Wei Shangguan [ctb], Rainer Petzold [ctb], Budiman Minasny [ctb], Bogdan Rosca [ctb], Nic Jelinski [ctb], Wiktor Zelazny [ctb], Rodolfo Marcondes Silva Souza [ctb], Jose Lucas Safanelli [ctb], Alexandre ten Caten [ctb]

TT.plot   *Plot soil texture triangles / diagrams.*

Description

Plot a soil texture triangle (also called soil texture diagrams, or soil texture ternary plots), with or without background soil texture classes boundaries, and with or without soil texture data points. The triangle geometry depends on the soil texture classification system chosen ('class.sys' argument) or on 'forcing' parameters (see below). Both the boundaries of the background texture classification system and the texture data points can be transformed from one particle size limits system to another (the particle size limits system of the plot). Default behaviour is no transformation (set 'css.transf' argument to TRUE to allow transformation). There are 3 different way to set the triangle geometry and characteristics (1) setting the 'class.sys' argument [lowest priority], (2) changing one or several values of the 'geo' list of arguments or (3) setting the corresponding arguments of TT.plot() [highest priority]. These arguments are "blr.clock", "tlr.an", "blr.tx", "text.sum", and "base.css.ps.lim". Different geometry arguments can be set at different levels (1, 2 or 3). Case (1) should be used when one wants to use the 'default' triangle geometry associated with a given texture classification system (chosen with the 'class.sys' argument). Case (2) should be used when TT.plot() has been called previously, with a call like geo <- TT.plot(), so the 'geo' object returned can be used for setting the geometry of a new texture triangle TT.plot(geo = geo) identical to the previous one. Case (3) should be used whenever the user wants to set the geometry of a texture triangle plot different from default values of the texture classification system chosen, and without re-using the geometry from a previous plot. ON DEFAULT VALUES OF TT.plot() ARGUMENTS? As TT.plot() shares its arguments with many other functions, their default value is not defined in TT.plot() source code, but rather in a dedicated list object called 'TT.par' and stored in the environment TT.env. The function TT.get() is used to retrieve the default value of the arguments defined in TT.par (see ?TT.get). For instance, to know the default value of 'class.sys', you can type TT.get("class.sys"). To set a different default value for a given argument in R, use TT.set() (see ?TT.set). For instance to change the default value of 'class.sys', type TT.set("class.sys" = "USDA.TT").

Usage

```
TT.plot(
  geo = NULL,
  tri.data = NULL,
  add = FALSE,
  css.names = NULL,
  z.name = NULL,
  main = NULL,
  blr.tx = NULL,
  css.lab = NULL,
  text.sum = NULL,
  base.css.ps.lim = NULL,
  tri.css.ps.lim = NULL,
  dat.css.ps.lim = NULL,
  css.transf = NULL,
  text.transf.fun = NULL,
```

```
        trsf.add.opt1 = NULL,
        trsf.add.opt2 = NULL,
        unit.ps = NULL,
        unit.tx = NULL,
        blr.clock = NULL,
        tlr.an = NULL,
        font = NULL,
        font.axis = NULL,
        font.lab = NULL,
        font.main = NULL,
        bg = NULL,
        fg = NULL,
        col = NULL,
        col.axis = NULL,
        col.lab = NULL,
        col.main = NULL,
        cex = NULL,
        cex.axis = NULL,
        cex.lab = NULL,
        cex.main = NULL,
        lwd = NULL,
        lwd.axis = NULL,
        lwd.lab = NULL,
        family.op = NULL,
        frame.bg.col = NULL,
        at = NULL,
        grid.show = NULL,
        grid.col = NULL,
        grid.lty = NULL,
        class.sys = NULL,
        class.lab.show = NULL,
        class.lab.col = NULL,
        class.line.col = NULL,
        class.p.bg.col = NULL,
        class.p.bg.hue = NULL,
        arrows.show = NULL,
        arrows.lty = NULL,
        points.type = NULL,
        pch = NULL,
        z.type = NULL,
        z.col.hue = NULL,
        z.cex.range = NULL,
        z.pch = NULL,
        text.tol = NULL,
        tri.sum.tst = NULL,
        tri.pos.tst = NULL,
        b.lim = NULL,
        l.lim = NULL,
```

```
    lang = NULL,
    new.mar = NULL,
    new.centroid = TRUE
)
```

Arguments

geo	List. 'geo' is one of the 3 way to set the texture triangle geometry. See there description and hierarchy in the function description. If geo != NULL, then geo must be a list containing 1 or several of the following items: "blr.clock", "tlr.an", "blr.tx", "text.sum", and "base.css.ps.lim". See the options with the same name for a description of the expected values and effects. The list can be created manually (like list("text.sum" = 1000)), or taken from the output of a previous call to TT.plot(), TT.baseplot() or TT.geo.get() (that return a 'geo' list).
tri.data	Data frame. Data frame containing the CLAY, SILT and SAND 'coordinates' of the texture data points to be plotted on top of the texture triangle and texture class boundaries. The data frame can contain more column than needed (ignored). The data frame must have column named CLAY, SILT and SAND (uppercase, the order has no importance) or named after the 'css.names' argument (alternative names). If 'z.name' argument is not NULL, the data frame must also contain a column named after 'z.name' value. The sum of CLAY, SILT and SAND must be equal to 'text.sum' ('text.tol' determines the error tolerance).
add	Single logical. If FALSE, a new plot is created. If FALSE, the plot is added to the existing one.
css.names	Vector of 3 character strings. Name of the columns in 'tri.data' that contains the CLAY SILT and SAND values, respectively. If NULL, default c("CLAY","SILT","SAND") value is assumed. Not to be confused with 'css.lab' that defines the labels of the CLAY SILT and SAND axes in the plot.
z.name	Single character string. Name of the column in 'tri.data' that contains the '4th quantitative variable' whose value must be used to define the points expansion factor and color (bubble plot). If NULL, a simple plot is drawn (no 'bubbles')
main	Single character string or expression. Main title of the plot.
blr.tx	Vector of 3 character strings. The 1st, 2nd and 3rd values must be either CLAY, SILT or SAND, and determines the particle size classes associated with the BOTTOM, LEFT and RIGHT axis, respectively. CLAY, SILT and SAND order in the vector is free, but they should all be used one time. The CLAY, SILT and SAND names must appear whatever the corresponding columns names in 'tri.data' (eventually set by 'css.names') and whatever the labels of the axis in the plot (eventually set by 'css.lab')
css.lab	Vector of 3 character strings or 3 expressions. The 1st, 2nd and 3rd values must be text strings or expressions, and determines the axes plot labels for the CLAY, SILT and SAND particle size classes, respectively. 'css.lab' values are independent from columns names in 'tri.data' (eventually set by 'css.names') and from whatever the placement of particle size classes on each axis (eventually set by 'blr.tx')
text.sum	Single numerical. Sum of the 3 particle size classes for each texture value (fixed). The real sum of the 3 particle size classes in 'tri.data' should be >=

text.sum * (1-text.tol) OR <= text.sum * (1+text.tol), where 'text.tol' is an argument that can be changed. If some of the texture values don't match this requirement, an error occur (function fails) and TT.plot returns a of bad values with their actual particle size classes sum. You can 'normalise' you data table () prior to the use of TT.plot, by using the function TT.normalise.sum(), so all values match the 'text.sum' criteria. See also 'tri.sum.tst' that can be set to FALSE to avoid sum of particle size classes tests.

base.css.ps.lim

Vector of 4 numericals. Particle size boundaries (upper and lower) of the 3 particle size classes (CLAY, SILT and SAND, starting from the lower size of CLAY particles, 0, to the upper size of the SAND particles, 2000), in micrometers, FOR THE BASE PLOT. These particles size class limits are the references and all other texture values with different limits will be converted into that reference if (and only if) css.transf == TRUE (not default). If NULL, 'base.css.ps.lim' will be set to the default value of the texture classification system chosen ('class.sys'). The transformation function is set by 'text.transf.fun' and is a log-linear interpolation by default.

tri.css.ps.lim Vector of 4 numericals. Particle size boundaries (upper and lower) of the 3 particle size classes (CLAY, SILT and SAND, starting from the lower size of CLAY particles, 0, to the upper size of the SAND particles, 2000), in micrometers, FOR THE TEXTURE TRIANGLE. If not NULL, different from 'base.css.ps.lim', and css.transf == TRUE (not default), then the CLAY SILT and SAND coordinates of the texture triangle will be converted into the 'base.css.ps.lim' reference. If NULL, 'tri.css.ps.lim' will be set to the default value of the texture classification system chosen ('class.sys'). The transformation function is set by 'text.transf.fun' and is a log-linear interpolation by default.

dat.css.ps.lim Vector of 4 numericals. Particle size boundaries (upper and lower) of the 3 particle size classes (CLAY, SILT and SAND, starting from the lower size of CLAY particles, 0, to the upper size of the SAND particles, 2000), in micrometers, FOR THE TEXTURE DATA TABLE ('tri.data'). If not NULL, different from 'base.css.ps.lim', and css.transf == TRUE (not default), then the CLAY SILT and SAND coordinates of the texture data in tri.data will be converted into the 'base.css.ps.lim' reference. If NULL, 'tri.css.ps.lim' will be set to the default value of the texture classification system chosen ('class.sys'). The transformation function is set by 'text.transf.fun' and is a log-linear interpolation by default.

css.transf Single logical. Set to TRUE to transform the texture coordinates of the texture triangle ('class.sys') or the texture data ('tri.data') into the base particle size class limits. See 'base.css.ps.lim' for the base plot particle size class limits, 'tri.css.ps.lim' for the triangle particle size class limits and 'dat.css.ps.lim' for the data table particle size class limits. The transformation function is set by 'text.transf.fun' and is a log-linear interpolation by default. The default value is FALSE, so no transformation is made.

text.transf.fun

R function with the same argument names and same output as the function TT.text.transf(). 'text.transf.fun' is the function that transform the texture values from one system of particle class size limits to another. Only used if css.transf

	== TRUE. Default value is text.transf.fun=TT.text.transf. See also 'base.css.ps.lim', 'tri.css.ps.lim' and 'dat.css.ps.lim'.
trsf.add.opt1	Non pre-defined format. If the user specifies its own texture transformation function in 'text.transf.fun' (not TT.text.transf()), then he can use 'trsf.add.opt1' and 'trsf.add.opt1' as new, additional, argument for his function. So the format of 'trsf.add.opt1' depends on the function defined by the user in 'text.transf.fun'.
trsf.add.opt2	Non pre-defined format. If the user specifies its own texture transformation function in 'text.transf.fun' (not TT.text.transf()), then he can use 'trsf.add.opt1' and 'trsf.add.opt1' as new, additional, argument for his function. So the format of 'trsf.add.opt1' depends on the function defined by the user in 'text.transf.fun'.
unit.ps	Single text string or expression. Unit of particle size class limits displayed on the plot (= part of the axis labels). Does not affect the other calculations. Default micrometers.
unit.tx	Single text string or expression. Unit of particle texture values displayed on the plot (= part of the axis labels). Does not affect the other calculations. Default is percentage.
blr.clock	Vector of logicals, eventually with NA values. Direction of increasing texture values on the BOTTOM, LEFT and RIGHT axis, respectively. A value of TRUE means that the axis direction is clockwise. A value of FALSE means that the axis direction is counterclockwise. A value of NA means that the axis direction is centripetal. Possible combinations are c(T,T,T); c(F,F,F); c(F,T,NA) and c(T,NA,F), for fully clockwise, fully counterclockwise, right centripetal and left centripetal orientations, respectively.
tlr.an	Vector of numericals. Value - in degrees - of the TOP, LEFT and RIGHT angles of the triangle. Any value between 0 and 90 is possible, but values belonging to 0 or 45 or 60 or 90 give a better graphical rendering.
font	Single integer. Not used yet.
font.axis	Single integer. Same definition as par("font.axis"). Font of the triangle axis's numbering.
font.lab	Single integer. Same definition as par("font.lab"). Font of the triangle axis's labels.
font.main	Single integer. Same definition as par("font.main"). Font of the triangle main title.
bg	Text string containing an R color code. Background color of the plot (= outside the triangle). See 'frame.bg.col' for the background color inside the triangle frame.
fg	Text string containing an R color code. DEPRECATED. foreground color of the plot (= point fill color).
col	Text string containing an R color code. Same definition as par("col"). Color of the points plotted in the triangle.
col.axis	Text string containing an R color code. Color of the triangle's axis (line and tics) The color of the texture classes boundaries is set by 'class.line.col'.
col.lab	Text string containing an R color code. Color of the triangle's labels (text) and arrows. The color of the texture classes labels is set by 'class.lab.col'.

col.main	Text string containing an R color code. Color of the main title.
cex	Vector of numerical or single numerical. Same definition as par("cex"). Expansion factor for the points plotted on the triangle.
cex.axis	Single numerical. Same definition as par("cex.axis"). Expansion factor for the axis tics labels (numbering).
cex.lab	Single numerical. Same definition as par("cex.lab"). Expansion factor for the axis labels AND the texture classes labels.
cex.main	Single numerical. Same definition as par("cex.main"). Expansion factor for the main title.
lwd	Single numerical. Same definition as par("lwd"). Line width for the graphical elements inside the triangle (points plotted).
lwd.axis	Single numerical. Same definition as par("lwd.axis"). Line width for the axis lines, tics and the grid lines inside the triangle.
lwd.lab	Single numerical. Same definition as par("lwd"). Line width for the direction arrows.
family.op	Single text string. Same definition as par("family"). Font type to be used in the plot text elements (title, labels)
frame.bg.col	Text string containing an R color code. Background color of the triangle plot (= inside the triangle). See 'bg' for the background color outside the triangle frame.
at	Vector of numericals. Location of the grid line start points on all 3 triangles axis. At the moment values are identical for all 3 axis, and changes to that parameter have not been tested.
grid.show	Single logical. If set to TRUE (the default) a grid is drawn on the background of the texture triangle. Set to FALSE to remove the grid.
grid.col	Text string containing an R color code. Color of the grid lines. If equal to NULL, then an appropriate color is used. Appropriate means (i) if 'class.p.bg.col' is FALSE (no color gradient in texture class polygons), then grid.col is equal to 'bg' (without transparency) unless a color is specified for the triangle frame background ('frame.bg.col'), in which case grid.col is a mix of 'frame.bg.col' and 'col.axis'. (ii) if 'class.p.bg.col' is TRUE, then grid.col is a light or dark color based on 'class.p.bg.hue' (light if 'bg' is dark and dark if 'bg' is light).
grid.lty	Single numerical. Line type of the grid lines (same types as par("lty")).
class.sys	Single text string. Text code of the texture classification system to be plotted on the background of the texture triangle. That texture classification system will determines the triangle geometry and particle class size system of the plot, unless higher level options are chosen (see the function definition). Possible values are "none" (no classification plotted), "USDA.TT" (USDA texture triangle), "HYPRES.TT" (texture triangle of the European Sil Map), "FR.AISNE.TT" (French texture triangle of the Aisne region soil survey), "FR.GEPPA.TT" (French GEPPA texture triangle), "DE.BK94.TT" (German texture triangle), "UK.SSEW.TT" (Soil Survey of England and Wales), "AU.TT" (Australian texture triangle), "BE.TT" (Belgium texture triangle), "CA.EN.TT" (Canadian texture triangle, with English class abbreviations) and "CA.FR.TT" (Canadian texture triangle, with French class abbreviations).

class.lab.show   Single text string. If equal to "abr" (default) or "full", labels are drawn inside texture class polygons with their full name ("full") or abbreviated name ("abr"). If equal to "none", no label is drawn.

class.lab.col   Text string containing an R color code. Color of the text label drawn inside texture class polygons.

class.line.col   Text string containing an R color code. Color of the texture class polygon boundary lines.

class.p.bg.col   Single logical OR vector of R colors (character strings). If FALSE (the default), no color gradient is used inside the texture class polygons. If TRUE, a color gradient is drawn, with the color hue specified in 'class.p.bg.hue' and with saturation and values that vary with texture. If 'class.p.bg.col' is a vector of R colors of the same length as the number of classes in the triangle, these colors will be used as background color for each texture classe plygons.

class.p.bg.hue   Single numerical. Only used if class.p.bg.col == TRUE (no default). Color hue (between 0 and 1) used to create a color gradient between the different texture class polygons.

arrows.show   Single logical. If TRUE (default), 3 arrows are drawn outside the triangle, along each axis, that show the direction of increasing values (arrow base) and of iso-value (arrow tip) of the texture class. If FALSE no arrows are drawn.

arrows.lty   Single numerical. Line type of the arrows drawn outside the triangle, along each axis. Same possible types as par("lty").

points.type   Single text letter. Point type. Either "p" (points only), "l" (lines only) or "b" (both points and lines), as for plot() or points(). Point refer here to soil texture values plotted on the triangle.

pch   Single numerical or vector of numericals, or single text string or vector of text string. Point shape number(s) or point character(s) to be plotted. Point refer here to soil texture values plotted on the triangle.

z.type   Single character string. Type of plot to be used for displaying a 4th variable on the texture triangle (in addition to Clay, Silt and Sand). Only used if 'z.name' is not NULL. Currently only one value is supported, "bubble", for displaying a bubble plot with bubble sizes and color saturation and values proportional to the value of tri.data[,z.name]. The value 'map' is deprecated and replaced by TT.iwd(), TT.image() or TT.contour().

z.col.hue   Single numerical. Hue of the bubble color ([0-1]) to be used if 'z.name' is not NULL. A gradient of saturation and value is automatically created for the bubbles (with this hue).

z.cex.range   Vector of 2 numericals. Minimum and maximum 'cex' of the bubbles plotted on the triangle if 'z.name' is not NULL.

z.pch   Single numerical or vector of numericals. Point symbol number(s) to be used for the bubbles if 'z.name' is not NULL.

text.tol   Single numerical. Tolerance on the sum of the 3 particle size classes. The real sum of the 3 particle size classes in 'tri.data' should be >= text.sum * (1-text.tol) OR <= text.sum * (1+text.tol). See 'text.sum' for more details, as well as 'tri.sum.tst' (to prevent texture sum tests).

tri.sum.tst	Single logical. If TRUE (the default), the sum of the 3 texture classes of each texture value in 'tri.data' will be checked in regard to 'text.sum' and 'text.tol'. If FALSE, no test is done.
tri.pos.tst	Single logical. If TRUE (the default), the position of texture values in 'tri.data' are tested to check that they are not OUTSIDE the texture triangle (i.e. that some texture values may be negative).
b.lim	Vector of 2 numerical values. This is an equivalent to plot() xlim argument. Minimum and maximum x / bottom value of the texture triangle area, in FRACTION OF THE MAXIMAL EXTENSION. Default is c(0,1). The real span is then b.lim * text.sum. This is a minimal 'zoom' implementation (results are not always perfect). 'b.lim' and 'l.lim' should be equal for better rendering.
l.lim	Vector of 2 numerical values. This is an equivalent to plot() ylim argument. Minimum and maximum y / left value of the texture triangle area, in FRACTION OF THE MAXIMAL EXTENSION. Default is c(0,1). The real span is then l.lim * text.sum. This is a minimal 'zoom' implementation (results are not always perfect). 'b.lim' and 'l.lim' should be equal for better rendering.
lang	Single text string. Determines the language used for the plot main title and axis labels. Possible values are 'en' (English, the default), "fr" (French), "it" (Italian), "es" (Spanish), "de" (German), "nl" (Dutch), "se" (Swedish) and "fl" (Flemish).
new.mar	Vector of 4 numericals. Margin sizes of the plot. Default is the same as par("mar"). See par("mar") for more details. Use this at your own risks!
new.centroid	Single logical. If TRUE (default) the new method (Paul Bourke) is used to calculate the centroid. If FALSE the centroid is taken as the mean x and y coordinates of the vertices.

Author(s)

Julien Moeys [aut, cre], Wei Shangguan [ctb], Rainer Petzold [ctb], Budiman Minasny [ctb], Bogdan Rosca [ctb], Nic Jelinski [ctb], Wiktor Zelazny [ctb], Rodolfo Marcondes Silva Souza [ctb], Jose Lucas Safanelli [ctb], Alexandre ten Caten [ctb]

Examples

```
require( soiltexture )

# ::: Texture triangles without data

# :: Base plot (HYPRES / European Soil Map triangle)
TT.plot()

# same as
TT.plot( class.sys = "HYPRES.TT" )

# :: Same plot, but with USDA texture triangle
TT.plot( class.sys = "USDA.TT" )

# :: Same plot, but with a color gradient
TT.plot(
```

```
        class.sys     = "USDA.TT",
        class.p.bg.col  = TRUE
    )   #

# :: No texture classification system
TT.plot( class.sys = "none" )

# ::: Texture triangles with texture data

# :: 1st create a dummy texture dataset
my.text <- data.frame(
    "CLAY"  = c(05,60,15,05,25,05,25,45,65,75,13,47),
    "SILT"  = c(05,08,15,25,55,85,65,45,15,15,17,43),
    "SAND"  = c(90,32,70,70,20,10,10,10,20,10,70,10),
    "OC"    = c(20,14,15,05,12,15,07,21,25,30,05,28)
)   #

# :: And plot it on a French Aisne texture triangle
#    with a title
TT.plot(
    class.sys   = "FR.AISNE.TT",
    tri.data    = my.text,
    main        = "Soil texture data"
)   #

# ::: Bubble plots (4th variable)

# :: 1st generate a dummy texture dataset with a 4th variable
#    with TT.dataset()
rand.text   <- TT.dataset( n = 100, seed.val = 1980042401 )

# :: Plot the dummy dataset as a bubble plot
TT.plot(
    class.sys   = "none",
    tri.data    = rand.text,
    z.name      = "Z",
    main        = "Soil texture triangle and Z bubble plot"
)   #

# ::: Test all the texture triangles
TT.plot( class.sys = "none" )          # no classification
TT.plot( class.sys = "HYPRES.TT" )     # HYPRES / European Soil Map
TT.plot( class.sys = "USDA.TT" )       # USDA
TT.plot( class.sys = "USDA-NCSS.TT" )  # USDA with NCSS labels
TT.plot( class.sys = "FR.AISNE.TT" )   # French Aisne
TT.plot( class.sys = "FR.GEPPA.TT" )   # French GEPPA
TT.plot( class.sys = "DE.BK94.TT" )    # Germany
TT.plot( class.sys = "DE.SEA74.TT" )   # German SEA 1974
TT.plot( class.sys = "DE.TGL85.TT" )   # German TGL 1985
TT.plot( class.sys = "UK.SSEW.TT" )    # UK
TT.plot( class.sys = "BE.TT" )         # Belgium
TT.plot( class.sys = "CA.FR.TT" )      # Canada (fr)
TT.plot( class.sys = "CA.EN.TT" )      # Canada (en)
```

```
TT.plot( class.sys = "CA2.EN.TT" )      # Canada (en) with official labels
TT.plot( class.sys = "AU2.TT" )         # Australian
TT.plot( class.sys = "ISSS.TT" )        # ISSS
TT.plot( class.sys = "ROM.TT" )         # Romanian
TT.plot( class.sys = "USDA1911" )       # USDA 1911 (M. Whitney, 1911)
TT.plot( class.sys = "BRASIL.TT" )      # Brasil (Lemos & Santos 1996)
TT.plot( class.sys = "SiBCS13.TT" )     # Brasil (Lemos & Santos 1996)


## Polish triangles:

#   PTG 1956-1959
try( TT.plot( class.sys = "PL.TT" ) )
#   PTG 1956 Musierowicz
try( TT.plot( class.sys = "Polish_PTG_1956_Musierowicz.TT" ) )
#   BN 1978
try( TT.plot( class.sys = "Polish_BN_1978.TT" ) )
#   PTG 2008
#   Polish kategorie agronomiczne 1990
try( TT.plot( class.sys =
    "Polish_kategorie_agronomiczne_1990.TT" ) )
try( TT.plot( class.sys = "PTG_2008.TT" ) )

#   All/most polish triangles includes special characters
#   and may not work on all platforms (or some accents may
#   be missing). Remove the try() around the code.

# ::: Test all the languages:
TT.plot( class.sys = "USDA.TT", lang = "en" )  # English, default
TT.plot( class.sys = "USDA.TT", lang = "fr" )  # French
TT.plot( class.sys = "USDA.TT", lang = "de" )  # German
TT.plot( class.sys = "USDA.TT", lang = "es" )  # Spanish
TT.plot( class.sys = "USDA.TT", lang = "it" )  # Italian
TT.plot( class.sys = "USDA.TT", lang = "nl" )  # Dutch
TT.plot( class.sys = "USDA.TT", lang = "fl" )  # Dutch (Belgium) / Flemish
TT.plot( class.sys = "USDA.TT", lang = "se" )  # Swedish
TT.plot( class.sys = "USDA.TT", lang = "ro" )  # Romanian

#   Languages with special characters
#   (may not work on all platforms + some accents can be missing)
try( TT.plot( class.sys = "USDA.TT", lang = "pl"  ) ) # Polish
try( TT.plot( class.sys = "USDA.TT", lang = "pt"  ) ) # Portuguese
try( TT.plot( class.sys = "USDA.TT", lang = "es2" ) ) # Spanish
try( TT.plot( class.sys = "USDA.TT", lang = "ro2" ) ) # Romanian
```

TT.points   *Plot a soil texture data table as points on an existing texture plot.*

Description

Plot a soil texture data table as points on an existing texture plot.

Usage

```
TT.points(
  tri.data,
  geo,
  css.names = NULL,
  z.name = NULL,
  base.css.ps.lim = NULL,
  dat.css.ps.lim = NULL,
  css.transf = NULL,
  text.transf.fun = NULL,
  trsf.add.opt1 = NULL,
  trsf.add.opt2 = NULL,
  text.tol = NULL,
  pch = NULL,
  fg = NULL,
  col = NULL,
  bg = NULL,
  cex = NULL,
  lwd = NULL,
  points.type = NULL,
  tri.sum.tst = NULL,
  tri.pos.tst = NULL,
  z.type = NULL,
  z.col.hue = NULL,
  z.cex.range = NULL,
  z.pch = NULL,
  text.sum = NULL,
  blr.clock = NULL,
  blr.tx = NULL
)
```

Arguments

`tri.data`	See `TT.plot`
`geo`	See `TT.plot`
`css.names`	See `TT.plot`
`z.name`	See `TT.plot`
`base.css.ps.lim`	
	See `TT.plot`
`dat.css.ps.lim`	See `TT.plot`
`css.transf`	See `TT.plot`
`text.transf.fun`	
	See `TT.plot`
`trsf.add.opt1`	See `TT.plot`
`trsf.add.opt2`	See `TT.plot`
`text.tol`	See `TT.plot`

pch	See [TT.plot](#)
fg	See [TT.plot](#)
col	See [TT.plot](#)
bg	See [TT.plot](#)
cex	See [TT.plot](#)
lwd	See [TT.plot](#)
points.type	See [TT.plot](#)
tri.sum.tst	See [TT.plot](#)
tri.pos.tst	See [TT.plot](#)
z.type	See [TT.plot](#)
z.col.hue	See [TT.plot](#)
z.cex.range	See [TT.plot](#)
z.pch	See [TT.plot](#)
text.sum	See [TT.plot](#)
blr.clock	See [TT.plot](#)
blr.tx	See [TT.plot](#)

Author(s)

Julien Moeys [aut, cre], Wei Shangguan [ctb], Rainer Petzold [ctb], Budiman Minasny [ctb], Bogdan Rosca [ctb], Nic Jelinski [ctb], Wiktor Zelazny [ctb], Rodolfo Marcondes Silva Souza [ctb], Jose Lucas Safanelli [ctb], Alexandre ten Caten [ctb]

TT.points.in.classes   *Classify a table of soil texture data according to a soil texture triangle.*

Description

The function calculate in which classe(s) of a texture triangle (classification system defined by 'class.sys') lies each soil sample (with texture data) in the table 'tri.data'. As a sample may lie inside a texture class, but also at the edge of 2 or more texture classes, the function does not only output one single texture class per sample. If 'PiC.type' is 'n' or 'l', it rather output a table where each column is a texture class and each row a texture sample, and yes / no information about the belonging of the sample to each texture class. Alternatively, If 'PiC.type' is 't'it will output a text string (per sample) containing all the texture classes to which that point belong. The texture data in 'tri.data' can be transformed into another particle size system prior to their classification if needed. See the options base.css.ps.lim, tri.css.ps.lim, dat.css.ps.lim, css.transf and text.transf.fun. ON DEFAULT VALUES OF TT.points.in.classes() ARGUMENTS? As TT.points.in.classes() shares its arguments with many other functions, their default value is not defined in TT.points.in.classes() source code, but rather in a dedicated list object called 'TT.par' and stored in the environment TT.env. The function TT.get() is used to retrieve the default value of the arguments defined in TT.par (see ?TT.get). For instance, to know the default value of 'class.sys', you can type TT.get("class.sys"). To set a different default value for a given argument in R, use TT.set() (see ?TT.set). For instance to change the default value of 'class.sys', type TT.set("class.sys" = "USDA.TT").

Usage

```
TT.points.in.classes(
  tri.data,
  class.sys = NULL,
  PiC.type = NULL,
  css.names = NULL,
  text.sum = NULL,
  base.css.ps.lim = NULL,
  tri.css.ps.lim = NULL,
  dat.css.ps.lim = NULL,
  css.transf = NULL,
  text.transf.fun = NULL,
  trsf.add.opt1 = NULL,
  trsf.add.opt2 = NULL,
  text.tol = NULL,
  tri.sum.tst = NULL,
  tri.pos.tst = NULL,
  collapse = NULL,
  texture2xy = FALSE,
  blr.tx = NULL,
  blr.clock = NULL
)
```

Arguments

tri.data	Data frame. Data frame containing the CLAY, SILT and SAND 'coordinates' of the texture data points to be classified The data frame can contain more column than needed (ignored). The data frame must have column named CLAY, SILT and SAND (uppercase, the order has no importance) or named after the 'css.names' argument (alternative names). The sum of CLAY, SILT and SAND must be equal to 'text.sum' ('text.tol' determines the error tolerance).
class.sys	Single text string. Text code of the texture classification system to be used for the classification of 'tri.data'. Possible values are "none" (no classification plotted), "USDA.TT" (USDA texture triangle), "HYPRES.TT" (texture triangle of the European Soil Map), "FR.AISNE.TT" (French texture triangle of the Aisne region soil survey), "FR.GEPPA.TT" (French GEPPA texture triangle), "DE.BK94.TT" (German texture triangle), "UK.SSEW.TT" (Soil Survey of England and Wales), "AU.TT" (Australian texture triangle), "BE.TT" (Belgium texture triangle), "CA.EN.TT" (Canadian texture triangle, with English class abbreviations) and "CA.FR.TT" (Canadian texture triangle, with French class abbreviations) (see the package vignette for a complete list).
PiC.type	Single character string. If equal to 'n', then a table of 0, 1, 2 or 3 is outputed (0 if the sample does not belong to a class, 1 if it does, 2 if it lies on an edge and 3 if it lies on a vertex). Notice that the accuracy of the classification is not garanteed for samples lying very close to an edge, or right on it. See <http://www.mail-archive.com/r-help@r-project.org/msg96180.html>
css.names	Vector of 3 character strings. Name of the columns in 'tri.data' that contains the CLAY SILT and SAND values, respectively. If NULL, default c("CLAY","SILT","SAND")

value is assumed. Not to be confused with 'css.lab' that defines the labels of the CLAY SILT and SAND axes in the plot.

text.sum   Single numerical. Sum of the 3 particle size classes for each texture value (fixed). The real sum of the 3 particle size classes in 'tri.data' should be >= text.sum * (1-text.tol) OR <= text.sum * (1+text.tol), where 'text.tol' is an argument that can be changed. If some of the texture values don't match this requirement, an error occur (function fails) and TT.points.in.classes returns a of bad values with their actual particle size classes sum. You can 'normalise' you data table () prior to the use of TT.points.in.classes, by using the function TT.normalise.sum(), so all values match the 'text.sum' criteria. See also 'tri.sum.tst' that can be set to FALSE to avoid sum of particle size classes tests.

base.css.ps.lim

Vector of 4 numericals. Particle size boundaries (upper and lower) of the 3 particle size classes (CLAY, SILT and SAND, starting from the lower size of CLAY particles, 0, to the upper size of the SAND particles, 2000), in micrometers, FOR THE BASE SYSTEM. These particles size class limits are the references and all other texture values with different limits will be converted into that reference if (and only if) css.transf == TRUE (not default). If NULL, 'base.css.ps.lim' will be set to the default value of the texture classification system chosen ('class.sys'). The transformation function is set by 'text.transf.fun' and is a log-linear interpolation by default.

tri.css.ps.lim   Vector of 4 numericals. Particle size boundaries (upper and lower) of the 3 particle size classes (CLAY, SILT and SAND, starting from the lower size of CLAY particles, 0, to the upper size of the SAND particles, 2000), in micrometers, FOR THE TEXTURE TRIANGLE. If not NULL, different from 'base.css.ps.lim', and css.transf == TRUE (not default), then the CLAY SILT and SAND coordinates of the texture triangle will be converted into the 'base.css.ps.lim' reference. If NULL, 'tri.css.ps.lim' will be set to the default value of the texture classification system chosen ('class.sys'). The transformation function is set by 'text.transf.fun' and is a log-linear interpolation by default.

dat.css.ps.lim   Vector of 4 numericals. Particle size boundaries (upper and lower) of the 3 particle size classes (CLAY, SILT and SAND, starting from the lower size of CLAY particles, 0, to the upper size of the SAND particles, 2000), in micrometers, FOR THE TEXTURE DATA TABLE ('tri.data'). If not NULL, different from 'base.css.ps.lim', and css.transf == TRUE (not default), then the CLAY SILT and SAND coordinates of the texture data in tri.data will be converted into the 'base.css.ps.lim' reference. If NULL, 'tri.css.ps.lim' will be set to the default value of the texture classification system chosen ('class.sys'). The transformation function is set by 'text.transf.fun' and is a log-linear interpolation by default.

css.transf   Single logical. Set to TRUE to transform the texture coordinates of the texture triangle ('class.sys') or the texture data ('tri.data') into the base particle size class limits. See 'base.css.ps.lim' for the base plot particle size class limits, 'tri.css.ps.lim' for the triangle particle size class limits and 'dat.css.ps.lim' for the data table particle size class limits. The transformation function is set by 'text.transf.fun' and is a log-linear interpolation by default. The default value is FALSE, so no transformation is made.

text.transf.fun

R function with the same argument names and same output as the function
TT.text.transf(). 'text.transf.fun' is the function that transform the texture values
from one system of particle class size limits to another. Only used if css.transf
== TRUE. Default value is text.transf.fun=TT.text.transf. See also 'base.css.ps.lim',
'tri.css.ps.lim' and 'dat.css.ps.lim'.

trsf.add.opt1

Non pre-defined format. If the user specifies its own texture transformation
function in 'text.transf.fun' (not TT.text.transf()), then he can use 'trsf.add.opt1'
and 'trsf.add.opt1' as new, additional, argument for his function. So the format
of 'trsf.add.opt1' depends on the function defined by the user in 'text.transf.fun'.

trsf.add.opt2

Non pre-defined format. If the user specifies its own texture transformation
function in 'text.transf.fun' (not TT.text.transf()), then he can use 'trsf.add.opt1'
and 'trsf.add.opt1' as new, additional, argument for his function. So the format
of 'trsf.add.opt1' depends on the function defined by the user in 'text.transf.fun'.

text.tol

Single numerical. Tolerance on the sum of the 3 particle size classes. The
real sum of the 3 particle size classes in 'tri.data' should be >= text.sum * (1-
text.tol) OR <= text.sum * (1+text.tol). See 'text.sum' for more details, as well
as 'tri.sum.tst' (to prevent texture sum tests).

tri.sum.tst

Single logical. If TRUE (the default), the sum of the 3 texture classes of each
texture value in 'tri.data' will be checked in regard to 'text.sum' and 'text.tol'.
If FALSE, no test is done.

tri.pos.tst

Single logical. If TRUE (the default), the position of texture values in 'tri.data'
are tested to check that they are not OUTSIDE the texture triangle (i.e. that some
texture values may be negative).

collapse

Single character string. If PiC.type = "t" and a sample lie on the edge of 2
texture classes, then both will be outputed in a single character string, separated
by 'collapse'. Example of output: [1] "C" "VF, F" "C" "C" "M"

texture2xy

Single logical. Set to FALSE to avoid any transformation of the texture data
(trigonometric) prior to testure data classification. Setting to FALSE avoid some
numerical accuracy problems when a point is on the border of a texture class.

blr.tx

Vector of 3 character strings. The 1st, 2nd and 3rd values must be either CLAY,
SILT or SAND, and determines the particle size classes associated with the
BOTTOM, LEFT and RIGHT axis, respectively. CLAY, SILT and SAND or-
der in the vector is free, but they should all be used one time. The CLAY, SILT
and SAND names must appear whatever the corresponding columns names in
'tri.data' (eventually set by 'css.names') and whatever the labels of the axis in
the plot (eventually set by 'css.lab')

blr.clock

Vector of logicals, eventually with NA values. Direction of increasing texture
values on the BOTTOM, LEFT and RIGHT axis, respectively. A value of TRUE
means that the axis direction is clockwise. A value of FALSE means that the
axis direction is counterclockwise. A value of NA means that the axis direc-
tion is centripetal. Possible combinations are c(T,T,T); c(F,F,F); c(F,T,NA) and
c(T,NA,F), for fully clockwise, fully counterclockwise, right centripetal and left
centripetal orientations, respectively.

Author(s)

Julien Moeys [aut, cre], Wei Shangguan [ctb], Rainer Petzold [ctb], Budiman Minasny [ctb], Bog-
dan Rosca [ctb], Nic Jelinski [ctb], Wiktor Zelazny [ctb], Rodolfo Marcondes Silva Souza [ctb],
Jose Lucas Safanelli [ctb], Alexandre ten Caten [ctb]

Examples

```
require( "soiltexture" )

# Create a dummy data frame of soil textures:
my.text <- data.frame(
    "CLAY"  = c(05,60,15,05,25,05,25,45,65,75,13,47),
    "SILT"  = c(05,08,15,25,55,85,65,45,15,15,17,43),
    "SAND"  = c(90,32,70,70,20,10,10,10,20,10,70,10),
    "OC"    = c(20,14,15,05,12,15,07,21,25,30,05,28)
)   #

# Display the table:
my.text

# Classify according to the HYPRES / European Soil Map classification
TT.points.in.classes(
    tri.data    = my.text[1:5,],
    class.sys   = "HYPRES.TT"
)   #

# Classify according to the USDA classification
TT.points.in.classes(
    tri.data    = my.text[1:5,],
    class.sys   = "USDA.TT"
)   #

# Classify according to the HYPRES / European Soil Map classification,
#   returns logical values
TT.points.in.classes(
    tri.data    = my.text[1:5,],
    class.sys   = "HYPRES.TT",
    PiC.type    = "l"
)   #

# Classify according to the HYPRES / European Soil Map classification,
#   returns text
TT.points.in.classes(
    tri.data    = my.text[1:5,],
    class.sys   = "HYPRES.TT",
    PiC.type    = "t"
)   #

# Classify according to the HYPRES / European Soil Map classification,
#   returns text,
#   custom class separator in case of points belonging to
#   several classes.
```

```
TT.points.in.classes(
    tri.data    = my.text[1:5,],
    class.sys   = "HYPRES.TT",
    PiC.type    = "t",
    collapse    = ";"
)   #
```

TT.set   *Function to change / set the default package parameters.*

Description

Function to change / set the default package parameters as they are stored in the list TT.par in the environment TT.env. Use this function to change some deafult parameters for all the current R cession. Many functions of soiltexture take some of their parameter values in TT.par.

Usage

```
TT.set(
  ...,
  reset = FALSE,
  par.list = "TT.par",
  bkp.par.list = "TT.par.bkp",
  par.env = TT.env
)
```

Arguments

`...`	List of parameters and value in the form "par.name1" = par.value1, "par.name2" = par.value2... List of parameters to change.
`reset`	Single logical. If set to TRUE the parameter list is reset to default
`par.list`	Single character. Name of the list containing the parameters
`bkp.par.list`	Single character. Name of the backuped list containing the default parameters
`par.env`	An R environment. Name of the environment containing the parameter lists (no quotes)

Author(s)

Julien Moeys [aut, cre], Wei Shangguan [ctb], Rainer Petzold [ctb], Budiman Minasny [ctb], Bogdan Rosca [ctb], Nic Jelinski [ctb], Wiktor Zelazny [ctb], Rodolfo Marcondes Silva Souza [ctb], Jose Lucas Safanelli [ctb], Alexandre ten Caten [ctb]

TT.str *Stretch or reshape the range of value of some data set.*

Description

Function to 'stretch' or reshape the range of value of some data set. Usefull for cex parameter in plot.

Usage

```
TT.str(x, str.min = 0, str.max = 1)
```

Arguments

x	Vector of numeric values.
str.min	Minimun value after stretching.
str.max	Maximun value after stretching.

Author(s)

Julien Moeys [aut, cre], Wei Shangguan [ctb], Rainer Petzold [ctb], Budiman Minasny [ctb], Bogdan Rosca [ctb], Nic Jelinski [ctb], Wiktor Zelazny [ctb], Rodolfo Marcondes Silva Souza [ctb], Jose Lucas Safanelli [ctb], Alexandre ten Caten [ctb]

TT.text *Plot text labels for each values of a soil texture data table on an existing texture plot.*

Description

Plot text labels for each values of a soil texture data table on an existing texture plot.

Usage

```
TT.text(
  tri.data,
  geo,
  labels = NULL,
  css.names = NULL,
  base.css.ps.lim = NULL,
  dat.css.ps.lim = NULL,
  css.transf = NULL,
  text.transf.fun = NULL,
  trsf.add.opt1 = NULL,
  trsf.add.opt2 = NULL,
```

```
    text.tol = NULL,
    text.sum = NULL,
    blr.clock = NULL,
    fg = NULL,
    col = NULL,
    cex = NULL,
    font = NULL,
    family.op = NULL,
    adj = NULL,
    pos = NULL,
    offset = NULL,
    tri.sum.tst = NULL,
    tri.pos.tst = NULL,
    blr.tx = NULL
)
```

Arguments

`tri.data`	See [TT.plot](#)
`geo`	See [TT.plot](#)
`labels`	Vector of character strings. Label to be plotted, for each point in 'tri.data'
`css.names`	See [TT.plot](#)
`base.css.ps.lim`	
	See [TT.plot](#)
`dat.css.ps.lim`	See [TT.plot](#)
`css.transf`	See [TT.plot](#)
`text.transf.fun`	
	See [TT.plot](#)
`trsf.add.opt1`	See [TT.plot](#)
`trsf.add.opt2`	See [TT.plot](#)
`text.tol`	See [TT.plot](#)
`text.sum`	See [TT.plot](#)
`blr.clock`	See [TT.plot](#)
`fg`	See [TT.plot](#)
`col`	See [text](#)
`cex`	See [text](#)
`font`	See [text](#)
`family.op`	See [TT.plot](#)
`adj`	See [text](#)
`pos`	See [text](#)
`offset`	See [text](#)
`tri.sum.tst`	See [TT.plot](#)
`tri.pos.tst`	See [TT.plot](#)
`blr.tx`	See [TT.plot](#)

Author(s)

Julien Moeys [aut, cre], Wei Shangguan [ctb], Rainer Petzold [ctb], Budiman Minasny [ctb], Bog-
dan Rosca [ctb], Nic Jelinski [ctb], Wiktor Zelazny [ctb], Rodolfo Marcondes Silva Souza [ctb],
Jose Lucas Safanelli [ctb], Alexandre ten Caten [ctb]

TT.text.transf	*Log-linear transformation of a soil texture data table between 2 parti-cle size systems (3 classes).*

Description

Log-linear transformation of a soil texture data table ('tri.data') from one particle size system
('dat.css.ps.lim') into another ('base.css.ps.lim'). Only 3 particle size classes allowed. See TT.text.transf.X
for transformation involving more than 3 particle classes. 'tri.data' may contain other variables (not
in 'css.names'). They are returned unchanged with the transformed texture data.

Usage

```
TT.text.transf(
  tri.data,
  base.css.ps.lim,
  dat.css.ps.lim,
  css.names = NULL,
  blr.tx = NULL,
  text.sum = NULL,
  text.tol = NULL,
  tri.sum.tst = NULL,
  tri.pos.tst = NULL,
  trsf.add.opt1 = NULL,
  trsf.add.opt2 = NULL
)
```

Arguments

tri.data	See TT.plot
base.css.ps.lim	
	See TT.plot
dat.css.ps.lim	See TT.plot
css.names	See TT.plot
blr.tx	See TT.plot
text.sum	See TT.plot
text.tol	See TT.plot
tri.sum.tst	See TT.plot
tri.pos.tst	See TT.plot
trsf.add.opt1	See TT.plot
trsf.add.opt2	See TT.plot

Author(s)

Julien Moeys [aut, cre], Wei Shangguan [ctb], Rainer Petzold [ctb], Budiman Minasny [ctb], Bogdan Rosca [ctb], Nic Jelinski [ctb], Wiktor Zelazny [ctb], Rodolfo Marcondes Silva Souza [ctb], Jose Lucas Safanelli [ctb], Alexandre ten Caten [ctb]

TT.text.transf.X   *Log-linear transformation of a soil texture data table between 2 particle size systems (X classes).*

Description

Log-linear transformation of a soil texture data table ('tri.data') from one particle size system ('dat.css.ps.lim') into another ('base.css.ps.lim'). No limit in the number of partile size classes in the inputed and outputed texture tables. See TT.text.transf for transformation involving only 3 particle classes. 'tri.data' can only contain texture data.

Usage

```
TT.text.transf.X(
  tri.data,
  base.ps.lim,
  dat.ps.lim,
  text.sum = NULL,
  text.tol = NULL,
  tri.sum.tst = NULL,
  tri.pos.tst = NULL
)
```

Arguments

tri.data	See TT.plot
base.ps.lim	Vector of numeric value. Particle size limits of the system tri.data should be converted into.
dat.ps.lim	Vector of numeric value. Particle size limits of tri.data.
text.sum	See TT.plot
text.tol	See See TT.plot
tri.sum.tst	See TT.plot
tri.pos.tst	See TT.plot

Author(s)

Julien Moeys [aut, cre], Wei Shangguan [ctb], Rainer Petzold [ctb], Budiman Minasny [ctb], Bogdan Rosca [ctb], Nic Jelinski [ctb], Wiktor Zelazny [ctb], Rodolfo Marcondes Silva Souza [ctb], Jose Lucas Safanelli [ctb], Alexandre ten Caten [ctb]

TT.vertices.plot   *Plot the vertices of a texture classification system.*

Description

Plot the vertices of a texture classification system, on top of an already drawn texture triangle plot. Also plot the vertices numbers. See TT.vertices.tbl() and TT.classes.tbl() for a non graphic, tabular equivalent of the plot.

Usage

```
TT.vertices.plot(
  geo,
  class.sys = "HYPRES.TT",
  fg = NULL,
  col = NULL,
  cex = NULL,
  font = NULL,
  family.op = NULL,
  adj = NULL,
  pos = NULL,
  offset = NULL,
  blr.tx = NULL,
  text.sum = NULL,
  blr.clock = NULL
)
```

Arguments

geo	See `TT.plot`
class.sys	See `TT.plot`
fg	See `TT.plot`
col	See `TT.plot`
cex	See `TT.plot`
font	See `TT.plot`
family.op	See `TT.plot`
adj	Passed to See `TT.text`
pos	Passed to See `TT.text`
offset	Passed to See `TT.text`
blr.tx	See `TT.plot`
text.sum	See `TT.plot`
blr.clock	See `TT.plot`

Author(s)

Julien Moeys [aut, cre], Wei Shangguan [ctb], Rainer Petzold [ctb], Budiman Minasny [ctb], Bog-
dan Rosca [ctb], Nic Jelinski [ctb], Wiktor Zelazny [ctb], Rodolfo Marcondes Silva Souza [ctb],
Jose Lucas Safanelli [ctb], Alexandre ten Caten [ctb]

TT.vertices.tbl   *Returns the table of vertices of a texture classification system.*

Description

Returns the table of vertices of a texture classification system. Returns the clay silt sand coordinates
of each vertices. Use TT.classes.tbl() to know the vertices that bounds each texture class. See also
TT.vertices.plot().

Usage

```
TT.vertices.tbl(class.sys = "HYPRES.TT")
```

Arguments

class.sys   See TT.plot

Author(s)

Julien Moeys [aut, cre], Wei Shangguan [ctb], Rainer Petzold [ctb], Budiman Minasny [ctb], Bog-
dan Rosca [ctb], Nic Jelinski [ctb], Wiktor Zelazny [ctb], Rodolfo Marcondes Silva Souza [ctb],
Jose Lucas Safanelli [ctb], Alexandre ten Caten [ctb]

Index