Package 'soiltexture'

March 17, 2011

Version 1.03

Date 2011-03-17

Title Functions for soil texture plot, classification and transformation

Author Julien MOEYS <jules_m78-soiltexture@yahoo.fr>, contributions from Wei Shangguan.

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

Depends R (>= 2.4.1), sp, MASS

Suggests drc

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. Several texture triangles are predefined: USDA; FAO (which is also the triangle for the soil map of Europe); Aisne (France); GEPPA (France); German triangle (Bodenkundliche Kartieranleitung 1994); Soil Survey of England and Wales (UK); Australian triangle; Belgian triangle; Canadian triangle; ISSS triangle; Romanian traingle. The Soil Texture Wizard is initially develloped from the functions 'soil.texture()' from the package 'plotrix' by Jim Lemon et al.

URL http://soiltexture.r-forge.r-project.org/

R topics documented:

soiltexture-package *Functions for soil texture plot, classification and transformation*

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. Several texture triangles are predefined: USDA; FAO (which is also the triangle for the soil map of Europe); Aisne (France); GEPPA (France); German triangle (Bodenkundliche Kartieranleitung 1994); Soil Survey of England and Wales (UK); Australian triangle; Belgian triangle; Canadian triangle; ISSS triangle; Romanian traingle. The Soil Texture Wizard is initially develloped from the functions 'soil.texture()' from the package 'plotrix' by Jim Lemon et al.

Details

Package:	soiltexture
Version:	1.03
Date:	2011-03-17
Title:	Functions for soil texture plot, classification and transformation
Author:	Julien MOEYS <jules_m78-soiltexture@yahoo.fr>, contributions from Wei Shangguan.
Maintainer:	Julien MOEYS <jules_m78-soiltexture@yahoo.fr>
Depends:	R (>= 2.4.1), sp, MASS
Suggests:	drc
License:	AGPL (>=3)
URL:	http://soiltexture.r-forge.r-project.org/

Author(s)

Julien MOEYS <jules_m78-soiltexture@yahoo.fr>

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

`...`

`par.list`

`bkp.par.list`

`par.env`

Author(s)

Julien MOEYS <jules_m78-soiltexture@yahoo.fr>

TT.auto.set — *Internal. Retrieve and set default values for parameters (par() or not), when NULL.*

Description

Retrieve and set default values for parameters (par() or not), when NULL.

Usage

```
TT.auto.set(fun = sys.function(which = -1), assign.op = TRUE, p.env = parent.fra
```

Arguments

`fun`

`assign.op`

`p.env`

`set.par`

Author(s)

Julien MOEYS <jules_m78-soiltexture@yahoo.fr>

TT.axis.arrows — *Internal. Plot the axis' arrows of a texture triangle plot.*

Description

Plot the axis' arrows of a texture triangle plot.

Usage

```
TT.axis.arrows(geo, css.lab = NULL, a.l = TT.get("arrows.lims"), a.h.s = TT.get(
```

Arguments

`geo`

`css.lab`

`a.l`

`a.h.s`

`a.t.s`

`a.t.s2`

`a.b.s`

`text.tol`

`text.sum`

`blr.clock`

`tlr.an`

`base.css.ps.lim`

`tri.sum.tst`

`tri.pos.tst`

`lwd.lab`

`arrows.lty`

`col.lab`

`font.lab`

`cex.lab`

`family.op`

`unit.ps`

`unit.tx`

`lang`

Author(s)

Julien MOEYS <jules_m78-soiltexture@yahoo.fr>

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
```

Arguments

`geo`

`class.sys`

`blr.clock`

`tlr.an`

`blr.tx`

`text.sum`
`base.css.ps.lim`

`tri.sum.tst`

`tri.pos.tst`

`text.tol`

`unit.ps`

`unit.tx`

`b.lim`

`l.lim`

`main`

`new.mar`

`bg`

`fg`

`col`

`cex.main`

`lang`

Author(s)

Julien MOEYS <jules_m78-soiltexture@yahoo.fr>

`TT.blr.ps.lim` *Internal. Create a tabular version of clay silt sand particle size limits.*

Description

Create a tabular version of clay silt sand particle size limits.

Usage

```
TT.blr.ps.lim(blr.tx, css.ps.lim)
```

Arguments

`blr.tx`

`css.ps.lim`

Author(s)

Julien MOEYS <jules_m78-soiltexture@yahoo.fr>

`TT.blr.tx.check` *Internal. Check the consistency between blr.tx and css.names.*

Description

Check the consistency between blr.tx and css.names. All values in blr.tx should be found in css.names and vice-versa.

Usage

```
TT.blr.tx.check(blr.tx, css.names)
```

Arguments

`blr.tx`	
`css.names`	

Author(s)

Julien MOEYS <jules_m78-soiltexture@yahoo.fr>

`TT.check.ps.lim` *Internal. Check the consistency between 'base.ps.lim' and 'dat.ps.lim'.*

Description

Check the consistency between 'base.ps.lim' and 'dat.ps.lim'. 5 tests performed.

Usage

```
TT.check.ps.lim(base.ps.lim, dat.ps.lim, ps.lim.length = c(4, 4))
```

Arguments

`base.ps.lim`	
`dat.ps.lim`	
`ps.lim.length`	vector of 2 integers. Number of particle size classes + 1. c(base,dat)

Author(s)

Julien MOEYS <jules_m78-soiltexture@yahoo.fr>

`TT.check.ps.lim.Xm` *Internal. Check the consistency between 'base.ps.lim' and 'dat.ps.lim'.*

Description

Check the consistency between 'base.ps.lim' and 'dat.ps.lim'. 4 tests performed.

Usage

```
TT.check.ps.lim.Xm(base.ps.lim, dat.ps.lim, ps.lim.length = c(4, 4))
```

Arguments

`base.ps.lim`	
`dat.ps.lim`	
`ps.lim.length`	vector of 2 integers. Number of particle size classes + 1. c(base,dat)

Author(s)

Wei Shangguan Julien MOEYS <jules_m78-soiltexture@yahoo.fr>

`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`

`divisorvar`

`css.names`

Author(s)

Julien MOEYS <jules_m78-soiltexture@yahoo.fr>

`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
```

Arguments

```
geo
class.sys
tri.css.ps.lim

css.transf
text.transf.fun

trsf.add.opt1

trsf.add.opt2

text.tol
text.sum
base.css.ps.lim

blr.tx
blr.clock
tri.sum.tst
tri.pos.tst
bg
class.lab.col

class.p.bg.col

class.p.bg.hue

class.line.col

class.lty
class.lab.show

cex.lab
font.lab
```

`family.op`	
`lwd.axis`	
`col.axis`	
`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 <jules_m78-soiltexture@yahoo.fr>

`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 = "FAO50.TT", collapse = ", ")
```

Arguments

`class.sys`
`collapse`

Author(s)

Julien MOEYS <jules_m78-soiltexture@yahoo.fr>

`TT.col2hsv` *Convert any colors to hsv.*

Description

Convert any colors to hsv. Wrapper around rgb2hsv() and col2rgb().

Usage

```
TT.col2hsv(col)
```

Arguments

`col`

Author(s)

Julien MOEYS <jules_m78-soiltexture@yahoo.fr>

`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
```

Arguments

```
geo
x
add
tri.sum.tst
tri.pos.tst
text.tol
unit.ps
unit.tx
b.lim
l.lim
main
new.mar
bg
fg
col
cex.main
lang
nlevels
levels
labels
xlim
ylim
zlim
labcex
drawlabels
method
axes
frame.plot
```

`lty`

`lwd`

`blr.clock`

`tlr.an`

`blr.tx`

`text.sum`
`base.css.ps.lim`

`...`

Author(s)

Julien MOEYS <jules_m78-soiltexture@yahoo.fr>

`TT.css2xy` *Converts texture data (3 classes) into x-y coordinates.*

Description

Converts texture data (3 classes) into x-y coordinates. This function is the 'heart' of most soiltexture plot functions.

Usage

```
TT.css2xy(tri.data, geo, css.names = NULL, text.tol = NULL, tri.sum.tst = NULL,
```

Arguments

`tri.data`

`geo`

`css.names`

`text.tol`

`tri.sum.tst`

`tri.pos.tst`

`set.par`

`text.sum`

`blr.clock`

Author(s)

Julien MOEYS <jules_m78-soiltexture@yahoo.fr>

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.s
```

Arguments

```
tri.data
css.names
text.sum
text.tol
tri.sum.tst
tri.pos.tst
```

Author(s)

Julien MOEYS <jules_m78-soiltexture@yahoo.fr>

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, t
```

Arguments

```
tri.data
text.sum
text.tol
tri.sum.tst
tri.pos.tst
```

Author(s)

Julien MOEYS <jules_m78-soiltexture@yahoo.fr>

`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`

`seed.val`

`css.names`

`text.sum`

Author(s)

Julien MOEYS <jules_m78-soiltexture@yahoo.fr>

`TT.deg2rad` *Function to convert angle in degree to angle in radian.*

Description

Function to convert angle in degree to angle in radian.

Usage

```
TT.deg2rad(A)
```

Arguments

`A`	Angle in Degrees

Author(s)

Julien MOEYS <jules_m78-soiltexture@yahoo.fr>

`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 <jules_m78-soiltexture@yahoo.fr>

`TT.DJ.col` *A function to obtaine a weight average 'mix' of different colors!*

Description

A function to obtaine a weight average 'mix' of different colors!

Usage

```
TT.DJ.col(cl, w, gray.l = FALSE)
```

Arguments

`cl`

`w`

`gray.l`

Author(s)

Julien MOEYS <jules_m78-soiltexture@yahoo.fr>

`TT.edges` *Internal. Plot the edges (bare axis) of a soil texture triangle.*

Description

Plot the edges (bare axis) of a soil texture triangle. This is not a primary plot function, TT.baseplot() must have been called before (usually inside TT.plot()).

Usage

```
TT.edges(geo, text.tol = NULL, text.sum = NULL, blr.clock = NULL, col.axis = NUL
```

Arguments

`geo`

`text.tol`

`text.sum`

`blr.clock`

`col.axis`

`plot.axis`

`frame.bg.col`

`lwd.axis`

`tri.sum.tst`

`tri.pos.tst`

`bg`

Author(s)

Julien MOEYS <jules_m78-soiltexture@yahoo.fr>

`TT.env` *TT env*

Description

Environment for storing, hiding and protecting internal variables and functions

Usage

```
TT.env
```

TT.gen.op.set — *Internal. Retrieve and set default values from options.*

Description

Retrieve and set default values from options (that do _not_ superseed par()).

Usage

```
TT.gen.op.set(param, assign.op = TRUE, p.env = parent.frame())
```

Arguments

```
param
assign.op
p.env
```

Author(s)

Julien MOEYS <jules_m78-soiltexture@yahoo.fr>

TT.geo.get — *Internal. Retrieve and return the geometrical parameters from a list of parameter 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, tex
```

Arguments

```
class.sys
blr.clock
tlr.an
blr.tx
text.sum
base.css.ps.lim
```

Author(s)

Julien MOEYS <jules_m78-soiltexture@yahoo.fr>

TT.geo.set — *Internal. Takes "geo" values and assign them individually in the parent function.*

Description

Takes "geo" values and assign them individually in the parent function.

Usage

```
TT.geo.set(geo, p.env = parent.frame())
```

Arguments

`geo`

`p.env`

Author(s)

Julien MOEYS <jules_m78-soiltexture@yahoo.fr>

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

`...`

`par.list`

`bkp.par.list`

`par.env`

Author(s)

Julien MOEYS <jules_m78-soiltexture@yahoo.fr>

`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
```

Arguments

`geo`

`at`

`text.tol`

`text.sum`

`blr.clock`

`grid.col`

`grid.lty`

`lwd.axis`

`tri.sum.tst`

`tri.pos.tst`

`class.p.bg.col`

`class.p.bg.hue`

`frame.bg.col`

`bg`

`col.axis`

Author(s)

Julien MOEYS <jules_m78-soiltexture@yahoo.fr>

`TT.ifelse` *Internal. Flexible version of ifelse.*

Description

Flexible version of ifelse.

Usage

```
TT.ifelse(test, yes, no)
```

Arguments

```
test
yes
no
```

Author(s)

Julien MOEYS <jules_m78-soiltexture@yahoo.fr>

`TT.image` *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() [to be written], before or after plot.

Usage

```
TT.image(geo, x, add = FALSE, tri.sum.tst = NULL, tri.pos.tst = NULL, text.tol =
```

Arguments

```
geo
x
add
tri.sum.tst
tri.pos.tst
text.tol
unit.ps
unit.tx
b.lim
l.lim
main
```

`new.mar`

`bg`

`fg`

`cex.main`

`lang`

`xlim`

`ylim`

`zlim`

`col`

`oldstyle`

`blr.clock`

`tlr.an`

`blr.tx`

`text.sum`
`base.css.ps.lim`

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

Author(s)

Julien MOEYS <jules_m78-soiltexture@yahoo.fr>

`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, tri.pol.data = NULL, text.tol =
```

Arguments

`tri.data`

`z.name`

`geo`

`css.names`

`tri.pol.data`

`text.tol`

`text.sum`

`blr.clock`

`tri.sum.tst`

`tri.pos.tst`

`set.par`

`n`

`lims`

`max.dist`

`q.max.dist`

`pow`

Author(s)

Julien MOEYS <jules_m78-soiltexture@yahoo.fr>

`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.
```

Arguments

`geo`

`tri.data`

`css.names`

`text.tol`

`text.sum`

`blr.clock`

`tri.sum.tst`

`tri.pos.tst`

`set.par`

`n`

`lims`

Author(s)

Julien MOEYS <jules_m78-soiltexture@yahoo.fr>

`TT.lines`	*Internal. Used to plot line elements of a texture plot axis, ticks, arrows, etc.*

Description

Used to plot line elements of a texture plot axis, ticks, arrows, etc.

Usage

```
TT.lines(geo = geo, at.1.s = TT.get("at"), at.2.s = 1 - TT.get("at"), at.3.s = 0
```

Arguments

```
geo
at.1.s
at.2.s
at.3.s
at.1.e
at.2.e
at.3.e
text.tol
text.sum
blr.clock
tri.sum.tst
tri.pos.tst
```

Author(s)

Julien MOEYS <jules_m78-soiltexture@yahoo.fr>

`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.t
```

Arguments

`geo`	
`css.names`	
`text.tol`	
`tri.sum.tst`	
`tri.pos.tst`	
`set.par`	
`n`	
`type`	
`...`	Further argumets passed to locator()

Author(s)

Julien MOEYS <jules_m78-soiltexture@yahoo.fr>

`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
```

Arguments

```
geo
tri.data
css.names
text.tol
text.sum
blr.clock
tri.sum.tst
tri.pos.tst
set.par
n
center
cov.mat
inverted
...
alr
divisorvar
```

Author(s)

Julien MOEYS <jules_m78-soiltexture@yahoo.fr>

`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, t
```

Arguments

```
tri.data
css.names
text.sum
text.tol
tri.pos.tst
residuals
```

Author(s)

Julien MOEYS <jules_m78-soiltexture@yahoo.fr>

`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 = NUL
```

Arguments

```
tri.data
text.sum
text.tol
tri.pos.tst
residuals
```

Author(s)

Julien MOEYS <jules_m78-soiltexture@yahoo.fr>

`TT.par.op.set` *Internal. Retrieve and set default values from options with default in "par()".*

Description

Retrieve and set default values from options with default in "par()"

Usage

```
TT.par.op.set(param, assign.op = TRUE, p.env = parent.frame())
```

Arguments

`param`

`assign.op`

`p.env`

Author(s)

Julien MOEYS <jules_m78-soiltexture@yahoo.fr>

`TT.phi2dia` *Convert a soil particle phi value into diameter dia [micro-meters].*

Description

Convert a soil particle phi value into diameter dia [micro-meters]. See also TT.dia2phi(). dia = (2^-phi)*1000. Not used by the package.

Usage

```
TT.phi2dia(phi)
```

Arguments

`phi`

Author(s)

Julien MOEYS <jules_m78-soiltexture@yahoo.fr>

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 = NUL
```

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), "FAO50.TT" (FAO texture triangle with a 50 microns silt-sand limit. DEFAULT VALUE), "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 <jules_m78-soiltexture@yahoo.fr>

Examples

```
require( soiltexture )

# ::: Texture triangles without data

# :: Base plot (FAO triangle)
TT.plot()

# same as
TT.plot( class.sys = "FAO50.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 = "FAO50.TT" )        # FAO
TT.plot( class.sys = "USDA.TT" )         # USDA
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 = "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 = "AU.TT" )           # Australian
TT.plot( class.sys = "ISSS.TT" )         # ISSS
TT.plot( class.sys = "ROM.TT" )          # Romanian

# ::: 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 = "se" )  #  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 (Belgian) / Flemmish
TT.plot( class.sys = "USDA.TT", lang = "se" )  #  Swedish
TT.plot( class.sys = "USDA.TT", lang = "ro" )  #  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
```

Arguments

```
tri.data
geo
css.names
z.name
base.css.ps.lim

dat.css.ps.lim


css.transf
text.transf.fun
```

`trsf.add.opt1`

`trsf.add.opt2`

`text.tol`
`pch`
`fg`
`col`
`bg`
`cex`
`lwd`
`points.type`
`tri.sum.tst`
`tri.pos.tst`
`z.type`
`z.col.hue`
`z.cex.range`
`z.pch`
`text.sum`
`blr.clock`
`blr.tx`

Author(s)

Julien MOEYS <jules_m78-soiltexture@yahoo.fr>

`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 = NU
```

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), "FAO50.TT" (FAO texture triangle with a 50 microns silt-sand limit. DEFAULT VALUE), "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 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')
`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.

Author(s)

Julien MOEYS <jules_m78-soiltexture@yahoo.fr>

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 FAO classification
TT.points.in.classes(
    tri.data    = my.text[1:5,],
    class.sys   = "FAO50.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 FAO classification, returns logicals
TT.points.in.classes(
    tri.data    = my.text[1:5,],
    class.sys   = "FAO50.TT",
    PiC.type    = "l"
)   #

# Classify according to the FAO classification, returns text
TT.points.in.classes(
    tri.data    = my.text[1:5,],
    class.sys   = "FAO50.TT",
    PiC.type    = "t"
)   #

# Classify according to the FAO 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   = "FAO50.TT",
    PiC.type    = "t",
    collapse    = ";"
)   #
```

`TT.polygon.area` *Determines the area of 1 polygon (in x-y coordinates).*

Description

Determines the area of 1 non-intersecting polygon (in x-y coordinates). Used by TT.polygon.centroids(). pol.x[1]:pol.y[1] is supposed different from pol.x[n]:pol.y[n] (i.e. the polygon is NOT closed). After "http://local.wasp.uwa.edu.au/~pbourke/geometry/polyarea/ Calculating The Area And Centroid Of A Polygon. Written by Paul Bourke, July 1988".

Usage

```
TT.polygon.area(pol.x, pol.y)
```

Arguments

`pol.x`	Vector of numericals. X coordinates of each vertices of the polygon.
`pol.y`	Vector of numericals. Y coordinates of each vertices of the polygon.

Value

Returns a single numerical: area of the polygon.

Author(s)

Julien MOEYS <jules_m78-soiltexture@yahoo.fr>

TT.polygon.centroids *Determines the centroid of 1 polygon (in x-y coordinates).*

Description

Determines the centroid of 1 non-intersecting polygon (in x-y coordinates). Used to determine the centroid of each texture class in the texture triangle onces its clay silt sand coordinates have been converted to x-y coordinates. pol.x[1]:pol.y[1] is supposed different from pol.x[n]:pol.y[n] (i.e. the polygon is NOT closed). After "http://local.wasp.uwa.edu.au/~pbourke/geometry/polyarea/ Calculating The Area And Centroid Of A Polygon. Written by Paul Bourke, July 1988".

Usage

```
TT.polygon.centroids(pol.x, pol.y)
```

Arguments

`pol.x`	Vector of numericals. X coordinates of each vertices of the polygon.
`pol.y`	Vector of numericals. Y coordinates of each vertices of the polygon.

Value

Returns a vector of 2 numericals: x and y coordinates of the polygon's centroid. Vector items are names "x" and "y".

Author(s)

Julien MOEYS <jules_m78-soiltexture@yahoo.fr>

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
```

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 <jules_m78-soiltexture@yahoo.fr>

`TT.str` *Internal. 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`

`str.min`

`str.max`

Author(s)

Julien MOEYS <jules_m78-soiltexture@yahoo.fr>

`TT.switch` *Internal. Used in the plot axis drawings.*

Description

Used in the plot axis drawings.

Usage

```
TT.switch(blr.clock = TT.get("blr.clock"), c1 = NA, c2 = NA, c3 = NA, c4 = NA, b
```

Arguments

`blr.clock`

`c1`

`c2`

`c3`

`c4`

`blr.order`

Author(s)

Julien MOEYS <jules_m78-soiltexture@yahoo.fr>

`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,
```

Arguments

`tri.data`

`geo`

`labels`

`css.names`
`base.css.ps.lim`

`dat.css.ps.lim`

`css.transf`
`text.transf.fun`

`trsf.add.opt1`

`trsf.add.opt2`

`text.tol`

`text.sum`

`blr.clock`

`fg`

`col`

`cex`

`font`

`family.op`

`adj`

`pos`

`offset`

`tri.sum.tst`

`tri.pos.tst`

`blr.tx`

Author(s)

Julien MOEYS <jules_m78-soiltexture@yahoo.fr>

`TT.text.transf` *Log-linear transformation of a soil texture data table between 2 particle 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.
```

Arguments

`tri.data`

`base.css.ps.lim`

`dat.css.ps.lim`

`css.names`

`blr.tx`

`text.sum`

`text.tol`

`tri.sum.tst`

`tri.pos.tst`

`trsf.add.opt1`

`trsf.add.opt2`

Author(s)

Julien MOEYS <jules_m78-soiltexture@yahoo.fr>

`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 =
```

Arguments

```
tri.data
base.ps.lim
dat.ps.lim
text.sum
text.tol
tri.sum.tst
tri.pos.tst
```

Author(s)

Julien MOEYS <jules_m78-soiltexture@yahoo.fr>

`TT.text.transf.Xm`	*Transformations of a soil texture data table between 2 particle size systems (X classes), various methods.*

Description

using various Particle Size Distribution (PSD) models including Anderson (AD), Fredlund4P (F4P), Fredlund3P (F3P), modified logistic growth (ML), Offset-Nonrenormalized Lognormal (ONL), Offset-Renormalized Lognormal (ORL), Skaggs (S), van Genuchten type(VG),van Genuchten modified(VGM), Weibull (W), Logarithm(L), Logistic growth (LG), Simple Lognormal (SL),Shiozawa and Compbell (SC). The performance of PSD models is influenced by many aspects like soil texture class, number and position (or closeness) of observation points, clay content etc. The latter four PSD models perform worse than the former ten. The AD, F4P, S, and W model is recommended for most of texture classes. And it will be even better to compare several different PSD models and using the results of the model with the minimum residual sum of squares (or other measures). Sometimes, the fitting will failed for the iteration is not converged or some errors happened. Transformation of a soil texture data table ('tri.data') from one particle size system ('dat.ps.lim') into another ('base.ps.lim'). No limit in the number of texture classes in the input and output texture tables. See TT.text.transf for transformation involving only 3 particle classes. 'tri.data' can only contain texture data. This function requires the "drc" package to be installed.

Usage

```
TT.text.transf.Xm(tri.data, base.ps.lim, dat.ps.lim, text.sum = NULL, text.tol =
```

Arguments

`tri.data`	
`base.ps.lim`	
`dat.ps.lim`	
`text.sum`	
`text.tol`	
`tri.sum.tst`	
`tri.pos.tst`	
`psdmodel`	
`omethod`	see optim for available methods, the default "all" is to run all methods and choose the best results with minimum residual sum of squares (RSS).
`tri.sum.norm`	

Author(s)

Wei Shangguan Julien MOEYS <jules_m78-soiltexture@yahoo.fr>

Examples

```
require( "soiltexture" )
require( "drc" )

my.text4 <- data.frame(
    "CLAY"  = c(05,60,15,05,25,05,25,45,65,75,13,47),
    "FSILT" = c(02,04,10,15,25,40,35,20,10,05,10,20),
    "CSILT" = c(03,04,05,10,30,45,30,25,05,10,07,23),
    "SAND"  = c(90,32,70,70,20,10,10,10,20,10,70,10)
)   #

TT.text.transf.Xm(
  tri.data    = my.text4,
  base.ps.lim = c(0,2,20,50,2000),
  dat.ps.lim  = c(0,2,20,63,2000),
  psdmodel    = "S"
)   #

# TT.text.transf.Xm(
#   tri.data    = my.text4,
#   base.ps.lim = c(0,1,50,2000),
#   dat.ps.lim  = c(0,2,30,60,2000),
#   psdmodel    = "AD",
#   omethod     = "Nelder-Mead"
# )
```

TT.ticks *Internal. Plot the axis' ticks of a texture triangle plot.*

Description

Plot the axis' ticks of a texture triangle plot.

Usage

```
TT.ticks(geo, at = NULL, text.tol = NULL, text.sum = NULL, blr.clock = NULL, tk.
```

Arguments

geo

at

text.tol

text.sum

blr.clock

tk.s

tri.sum.tst

tri.pos.tst

lwd.axis

col.axis

Author(s)

Julien MOEYS <jules_m78-soiltexture@yahoo.fr>

TT.ticks.lab *Internal. Plot the axis ticks' labels of a texture triangle plot.*

Description

Plot the axis ticks' labels of a texture triangle plot.

Usage

```
TT.ticks.lab(geo, at = NULL, text.tol = NULL, text.sum = NULL, blr.clock = NULL,
```

Arguments

`geo`
`at`
`text.tol`
`text.sum`
`blr.clock`
`tlr.an`
`tk.ls`
`tri.sum.tst`
`tri.pos.tst`
`col.axis`
`font.axis`
`cex.axis`
`family.op`

Author(s)

Julien MOEYS <jules_m78-soiltexture@yahoo.fr>

`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 = "FAO50.TT", fg = NULL, col = NULL, cex = NULL,
```

Arguments

`geo`
`class.sys`
`fg`
`col`
`cex`
`font`
`family.op`
`adj`
`pos`
`offset`
`blr.tx`
`text.sum`
`blr.clock`

Author(s)

Julien MOEYS <jules_m78-soiltexture@yahoo.fr>

`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 = "FAO50.TT")
```

Arguments

`class.sys`

Author(s)

Julien MOEYS <jules_m78-soiltexture@yahoo.fr>

`TT.xy.grid` *Internal. Create a grid in the x-y coordinate system.*

Description

Create a grid in the x-y coordinate system. Most of the function is a reshaped extract from kde2d() from the MASS package, by Venables & Ripley (+ modifications)

Usage

```
TT.xy.grid(x, y, n = 25)
```

Arguments

`x`

`y`

`n`

Author(s)

Julien MOEYS <jules_m78-soiltexture@yahoo.fr>

`TT.xy2css` *Internal. Convert point-data duplets (2 variables, x-y coordinaes) in Clay silta and sand coordinates.*

Description

Internal. Convert point-data duplets (2 variables, x-y coordinaes) in Clay silta and sand coordinates.

Usage

```
TT.xy2css(xy.data, geo, css.names = NULL, text.tol = NULL, tri.sum.tst = NULL, t
```

Arguments

`xy.data`	a data.frame with xpos and ypos columns
`geo`	
`css.names`	
`text.tol`	
`tri.sum.tst`	
`tri.pos.tst`	
`set.par`	
`blr.clock`	
`text.sum`	

Author(s)

Julien MOEYS <jules_m78-soiltexture@yahoo.fr>

Index