

# Package ‘soiltexture’

October 14, 2022

**Version** 1.5.1

**Date** 2018-09-19

**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 (>= 3.1.0)

**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)

**URL** <https://github.com/julienmoeys/soiltexture>

**BugReports** <https://github.com/julienmoeys/soiltexture/issues>

**Imports** sp, MASS, tools, tcltk, utils

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2018-09-20 14:40:02 UTC

**R topics documented:**

soiltexture-package	3
soiltextureInfo	6
soiltexture_gui	8
TT.add	11
TT.auto.set	12
TT.axis.arrows	12
TT.baseplot	14
TT.blr.ps.lim	15
TT.blr.tx.check	16
TT.check.ps.lim	16
TT.chemometrics.alr	17
TT.classes	18
TT.classes.tbl	19
TT.col2hsv	20
TT.contour	20
TT.css2xy	22
TT.data.test	23
TT.data.test.X	24
TT.dataset	25
TT.deg2rad	25
TT.dia2phi	26
TT.DJ.col	26
TT.edges	27
TT.env	28
TT.gen.op.set	28
TT.geo.get	29
TT.geo.set	29
TT.get	30
TT.grid	31
TT.ifelse	32
TT.image	32
TT.iwd	34
TT.kde2d	35
TT.lines	36
TT.locator	37
TT.mahalanobis	38
TT.normalise.sum	40
TT.normalise.sum.X	40
TT.par.op.set	41
TT.phi2dia	42
TT.plot	42
TT.points	58
TT.points.in.classes	60
TT.polygon.area	64
TT.polygon.centroids	65
TT.set	66

TT.str . . . . .	67
TT.switch . . . . .	67
TT.text . . . . .	68
TT.text.transf . . . . .	70
TT.text.transf.X . . . . .	71
TT.ticks . . . . .	72
TT.ticks.lab . . . . .	73
TT.vertices.plot . . . . .	74
TT.vertices.tbl . . . . .	75
TT.xy.grid . . . . .	75
TT.xy2css . . . . .	76

<b>Index</b>	<b>77</b>
--------------	-----------

---

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. 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()`.

## Details

The DESCRIPTION file:

```

Package:      soiltexture
Version:      1.5.1
Date:         2018-09-19
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]
Authors@R:   c( person( "Julien", "Moeys", role = c("aut", "cre"), email = "jules_m78-soiltexture@yahoo.fr" ), person( "Wei Shangguan", "Wei Shangguan", role = c("ctb"), email = "shangguan@163.com" ), person( "Rainer Petzold", "Rainer Petzold", role = c("ctb"), email = "rainer.petzold@univie.ac.at" ), person( "Budiman Minasny", "Budiman Minasny", role = c("ctb"), email = "budiman.minasny@univie.ac.at" ), person( "Bogdan Rosca", "Bogdan Rosca", role = c("ctb"), email = "bogdan.rosca@univie.ac.at" ) )
Maintainer:  Julien Moeys <jules_m78-soiltexture@yahoo.fr>
Depends:     R (>= 3.1.0)
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.
License:     AGPL (>=3)
URL:         https://github.com/julienmoeys/soiltexture
BugReports:  https://github.com/julienmoeys/soiltexture/issues
Imports:     sp, MASS, tools, tcltk, utils

```

Index of help topics:

TT.DJ.col	Internal. A function to obtaine a weight average 'mix' of different colors!
TT.add	Function to add a new default package parameters.
TT.auto.set	Internal. Retrieve and set default values for parameters (par() or not), when NULL.
TT.axis.arrows	Internal. Plot the axis' arrows of a texture triangle plot.
TT.baseplot	Internal. Create an empty plot scene for a texture triangle.
TT.blr.ps.lim	Internal. Create a tabular version of clay silt sand particle size limits.
TT.blr.tx.check	Internal. Check the consistency between blr.tx and css.names.
TT.check.ps.lim	Internal. Check the consistency between 'base.ps.lim' and 'dat.ps.lim'.
TT.chemometrics.alr	Compute the additive log-ratio transformation of compositional data.
TT.classes	Plot the texture classes polygons in a texture triangle plot.
TT.classes.tbl	Returns the table of classes of a texture classification system.
TT.col2hsv	Internal. Convert any colors to hsv.
TT.contour	Wrapper for the contour() function adapted to texture triangles.
TT.css2xy	Internal. Converts texture data (3 classes) into x-y coordinates.
TT.data.test	Test the validity of some soil texture data table (3 particle size classes).
TT.data.test.X	Test the validity of some soil texture data table (X particle size classes).
TT.dataset	Genetates a virtual cross correlated clay silt sand + Z dataset.
TT.deg2rad	Internal. Function to convert angle in degree to angle in radian.
TT.dia2phi	Internal. Convert a soil particle diameter dia [micro-meters] into $\phi = -\log_2(\text{dia}/1000)$
TT.edges	Internal. Plot the edges (bare axis) of a soil texture triangle.
TT.env	TT env
TT.gen.op.set	Internal. Retrieve and set default values from options.
TT.geo.get	Internal. Retrieve and return the geometrical parameters from a list of parameter values (NULL or not).
TT.geo.set	Internal. Takes "geo" values and assign them individually in the parent function.
TT.get	Function to retrieve / get the default package

	parameters.
TT.grid	Plot a grid at regular texture intervals inside an existing soil texture triangle.
TT.ifelse	Internal. Flexible version of ifelse.
TT.image	Wrapper for the contour() function adapted to texture triangles.
TT.iwd	Inverse weighted distance interpolation on a grid.
TT.kde2d	Calculated the 2D probability density on an x-y grid.
TT.lines	Internal. Used to plot line elements of a texture plot axis, ticks, arrows, etc.
TT.locator	Interactive (mouse clic) retrieval the CLAY SILT SAND coordinate of points on a texture triangle.
TT.mahalanobis	Calculates the Mahalanobis distance between clay silt and sand.
TT.normalise.sum	Normalises the sum of the 3 particle size classes.
TT.normalise.sum.X	Normalises the sum of the X particle size classes.
TT.par.op.set	Internal. Retrieve and set default values from options with default in "par()".
TT.phi2dia	Internal. Convert a soil particle phi value into diameter dia [micro-meters].
TT.plot	Plot soil texture triangles / diagrams.
TT.points	Plot a soil texture data table as points on an existing texture plot.
TT.points.in.classes	Classify a table of soil texture data according to a soil texture triangle.
TT.polygon.area	Internal. Determines the area of 1 polygon (in x-y coordinates).
TT.polygon.centroids	Internal. Determines the centroid of 1 polygon (in x-y coordinates).
TT.set	Function to change / set the default package parameters.
TT.str	Internal. Stretch or reshape the range of value of some data set.
TT.switch	Internal. Used in the plot axis drawings.
TT.text	Plot text labels for each values of a soil texture data table on an existing texture plot.
TT.text.transf	Log-linear transformation of a soil texture data table between 2 particle size systems (3 classes).
TT.text.transf.X	Log-linear transformation of a soil texture data table between 2 particle size systems (X classes).
TT.ticks	Internal. Plot the axis' ticks of a texture

	triangle plot.
TT.ticks.lab	Internal. Plot the axis ticks' labels of a texture triangle plot.
TT.vertices.plot	Internal. Plot the vertices of a texture classification system.
TT.vertices.tbl	Returns the table of vertices of a texture classification system.
TT.xy.grid	Internal. Create a grid in the x-y coordinate system.
TT.xy2css	Internal. Convert point-data duplets (2 variables, x-y coordinaes) in Clay silta and sand coordinates.
soiltexture-package	Functions for Soil Texture Plot, Classification and Transformation
soiltextureInfo	Display and / or export system and package version information
soiltexture_gui	Text-based menu for plotting and classifying soil texture data

Further information is available in the following vignettes:

soiltexture\_vignette    The soil texture wizard: a tutorial (source)

### 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]

Maintainer: Julien Moeys <jules\_m78-soiltexture@yahoo.fr>

---

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" )
```

**Description**

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

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).

### 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 <a href="#">TT.plot</a> .
graphics	See <a href="#">select.list</a> .
...	Additional parameters passed to <code>soiltexture:::read.table.menu</code> or (subsequently) to <a href="#">read.table</a> .

### 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**

```
...
par.list
bkp.par.list
par.env
```

### **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.auto.set	<i>Internal. Retrieve and set default values for parameters (par() or not), when NULL.</i>
-------------	--

---

**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.frame(), set.par = TRUE)
```

**Arguments**

```
fun
assign.op
p.env
set.par
```

**Author(s)**

Julien Moeys [aut, cre], Wei Shanguan [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.axis.arrows	<i>Internal. Plot the axis' arrows of a texture triangle plot.</i>
----------------	--

---

**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("arrows.head.shift"), a.t.s = TT.get("arrows.text.shift"),
```

```
    a.t.s2 = TT.get("arrows.text.shift2"), a.b.s = TT.get("arrows.base.shift"),
```

```
text.tol = NULL, text.sum = NULL, blr.clock = NULL,  
  
t1r.an = NULL, base.css.ps.lim = NULL, tri.sum.tst = FALSE,  
  
tri.pos.tst = FALSE, lwd.lab = NULL, arrows.lty = NULL,  
  
col.lab = NULL, font.lab = NULL, cex.lab = NULL,  
  
family.op = NULL, unit.ps = NULL, unit.tx = NULL,  
  
lang = NULL)
```

**Arguments**

```
geo  
css.lab  
a.l  
a.h.s  
a.t.s  
a.t.s2  
a.b.s  
text.tol  
text.sum  
blr.clock  
t1r.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 [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  
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 [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.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 [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.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 [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.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 [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  
divisorvar  
css.names

**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 polygons 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`  
`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 [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  
 collapse

### 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.col2hsv                      *Internal. 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 [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  
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 [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.css2xy

*Internal. 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, tri.pos.tst = NULL, set.par = FALSE,

          text.sum = NULL, blr.clock = 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 [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  $\geq 0$ , (5) That the sum of the 3 particle size classes is  $\geq \text{'text.sum'} * (1 - \text{'text.tol'})$  or  $\leq \text{'text.sum'} * (1 + \text{'text.tol'})$ . '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  
css.names  
text.sum  
text.tol  
tri.sum.tst  
tri.pos.tst

**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  $\geq 0$ , (5) That the sum of the X particle size class is  $\geq \text{'text.sum'} * (1 - \text{'text.tol'})$  or  $\leq \text{'text.sum'} * (1 + \text{'text.tol'})$ . 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  
text.sum  
text.tol  
tri.sum.tst  
tri.pos.tst
```

**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	<i>Genetates a virtual cross correlated clay silt sand + Z dataset.</i>
------------	---

---

**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 [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.deg2rad	<i>Internal. Function to convert angle in degree to angle in radian.</i>
------------	--

---

**Description**

Function to convert angle in degree to angle in radian.

**Usage**

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

**Arguments**

A                    Angle in Degrees

**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                      *Internal. 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.DJ.col                      *Internal. 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 [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.edges	<i>Internal. Plot the edges (bare axis) of a soil texture triangle.</i>
----------	---

---

### 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 = NULL, plot.axis = TRUE, frame.bg.col = NULL,  
  
         lwd.axis = NULL, tri.sum.tst = NULL, tri.pos.tst = NULL,  
  
         bg = NULL)
```

### 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 [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.env	<i>TT env</i>
--------	---------------

---

**Description**

Environment for storing, hiding and protecting internal variables and functions

**Usage**

TT.env

---

TT.gen.op.set	<i>Internal. Retrieve and set default values from options.</i>
---------------	--

---

**Description**

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

**Usage**

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

**Arguments**

param

assign.op

p.env

**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	<i>Internal. Retrieve and return the geometrical parameters from a list of parameter values (NULL or not).</i>
------------	--

---

**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  
blr.clock  
tlr.an  
blr.tx  
text.sum  
base.css.ps.lim

**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.set	<i>Internal. Takes "geo" values and assign them individually in the parent function.</i>
------------	--

---

**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 [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**

...  
par.list  
bkp.par.list  
par.env

**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	<i>Plot a grid at regular texture intervals inside an existing soil texture triangle.</i>
---------	---

---

**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
- 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 [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.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 [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 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 = 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
- 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 [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.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 = 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  
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 [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 probability density on an x-y grid.*

---

**Description**

Function that calculated the 2D probability 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  
tri.data  
css.names  
text.tol  
text.sum  
blr.clock  
tri.sum.tst  
tri.pos.tst  
set.par  
n  
lims

**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.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, at.1.e = TT.get("at"),  
  
        at.2.e = 0, at.3.e = 1 - TT.get("at"), text.tol = NULL,  
  
        text.sum = NULL, blr.clock = NULL, tri.sum.tst = NULL,  
  
        tri.pos.tst = NULL)
```

**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 [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  
css.names  
text.tol  
tri.sum.tst  
tri.pos.tst  
set.par  
n  
type  
...           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  
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 [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      *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  
css.names  
text.sum  
text.tol  
tri.pos.tst  
residuals

**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  
text.sum  
text.tol  
tri.pos.tst  
residuals

**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.par.op.set	<i>Internal. Retrieve and set default values from options with default in "par()".</i>
---------------	--

---

**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 [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.phi2dia	<i>Internal. Convert a soil particle phi value into diameter dia [micro-meters].</i>
------------	--

---

### 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 [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	<i>Plot soil texture triangles / diagrams.</i>
---------	--

---

### 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.to1 = NULL, tri.sum.tst = NULL,  
  
tri.pos.tst = NULL, b.lim = NULL, l.lim = NULL,  
  
lang = NULL, new.mar = NULL, new.centroid = TRUE)
```

**Arguments**

<code>geo</code>	List. 'geo' is one of the 3 way to set the texture triangle geometry. See there description and hierarchy in the function description. If <code>geo != NULL</code> , then <code>geo</code> 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 <code>list("text.sum" = 1000)</code> ), or taken from the output of a previous call to <code>TT.plot()</code> , <code>TT.baseplot()</code> or <code>TT.geo.get()</code> (that return a 'geo' list).
<code>tri.data</code>	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).
<code>add</code>	Single logical. If FALSE, a new plot is created. If TRUE, the plot is added to the existing one.
<code>css.names</code>	Vector of 3 character strings. Name of the columns in 'tri.data' that contains the CLAY SILT and SAND values, respectively. If NULL, default <code>c("CLAY","SILT","SAND")</code> value is assumed. Not to be confused with 'css.lab' that defines the labels of the CLAY SILT and SAND axes in the plot.
<code>z.name</code>	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')
<code>main</code>	Single character string or expression. Main title of the plot.
<code>blr.tx</code>	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 $\geq \text{text.sum} * (1 - \text{text.tol})$ OR $\leq \text{text.sum} * (1 + \text{text.tol})$ , 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.

<code>dat.css.ps.lim</code>	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 <code>css.transf == TRUE</code> (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.
<code>css.transf</code>	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.
<code>text.transf.fun</code>	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 <code>css.transf == TRUE</code> . Default value is <code>text.transf.fun=TT.text.transf</code> . See also 'base.css.ps.lim', 'tri.css.ps.lim' and 'dat.css.ps.lim'.
<code>trsf.add.opt1</code>	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'.
<code>trsf.add.opt2</code>	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'.
<code>unit.ps</code>	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.

<code>unit.tx</code>	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.
<code>blr.clock</code>	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 <code>c(T,T,T)</code> ; <code>c(F,F,F)</code> ; <code>c(F,T,NA)</code> and <code>c(T,NA,F)</code> , for fully clockwise, fully counterclockwise, right centripetal and left centripetal orientations, respectively.
<code>tlr.an</code>	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.
<code>font</code>	Single integer. Not used yet.
<code>font.axis</code>	Single integer. Same definition as <code>par("font.axis")</code> . Font of the triangle axis's numbering.
<code>font.lab</code>	Single integer. Same definition as <code>par("font.lab")</code> . Font of the triangle axis's labels.
<code>font.main</code>	Single integer. Same definition as <code>par("font.main")</code> . Font of the triangle main title.
<code>bg</code>	Text string containing an R color code. Background color of the plot (= outside the triangle). See <code>'frame.bg.col'</code> for the background color inside the triangle frame.
<code>fg</code>	Text string containing an R color code. DEPRECATED. foreground color of the plot (= point fill color).
<code>col</code>	Text string containing an R color code. Same definition as <code>par("col")</code> . Color of the points plotted in the triangle.
<code>col.axis</code>	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 <code>'class.line.col'</code> .
<code>col.lab</code>	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 <code>'class.lab.col'</code> .
<code>col.main</code>	Text string containing an R color code. Color of the main title.
<code>cex</code>	Vector of numerical or single numerical. Same definition as <code>par("cex")</code> . Expansion factor for the points plotted on the triangle.
<code>cex.axis</code>	Single numerical. Same definition as <code>par("cex.axis")</code> . Expansion factor for the axis tics labels (numbering).

<code>cex.lab</code>	Single numerical. Same definition as <code>par("cex.lab")</code> . Expansion factor for the axis labels AND the texture classes labels.
<code>cex.main</code>	Single numerical. Same definition as <code>par("cex.main")</code> . Expansion factor for the main title.
<code>lwd</code>	Single numerical. Same definition as <code>par("lwd")</code> . Line width for the graphical elements inside the triangle (points plotted).
<code>lwd.axis</code>	Single numerical. Same definition as <code>par("lwd.axis")</code> . Line width for the axis lines, tics and the grid lines inside the triangle.
<code>lwd.lab</code>	Single numerical. Same definition as <code>par("lwd")</code> . Line width for the direction arrows.
<code>family.op</code>	Single text string. Same definition as <code>par("family")</code> . Font type to be used in the plot text elements (title, labels)
<code>frame.bg.col</code>	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.
<code>at</code>	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.
<code>grid.show</code>	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.
<code>grid.col</code>	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 <code>grid.col</code> is equal to 'bg' (without transparency) unless a color is specified for the triangle frame background ('frame.bg.col'), in which case <code>grid.col</code> is a mix of 'frame.bg.col' and 'col.axis'. (ii) if 'class.p.bg.col' is TRUE, then <code>grid.col</code> is a light or dark color based on 'class.p.bg.hue' (light if 'bg' is dark and dark if 'bg' is light).
<code>grid.lty</code>	Single numerical. Line type of the grid lines (same types as <code>par("lty")</code> ).
<code>class.sys</code>	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).
<code>class.lab.show</code>	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.
<code>class.lab.col</code>	Text string containing an R color code. Color of the text label drawn inside texture class polygons.
<code>class.line.col</code>	Text string containing an R color code. Color of the texture class polygon boundary lines.
<code>class.p.bg.col</code>	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.
<code>class.p.bg.hue</code>	Single numerical. Only used if <code>class.p.bg.col == TRUE</code> (no default). Color hue (between 0 and 1) used to create a color gradient between the different texture class polygons.
<code>arrows.show</code>	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 isovalue (arrow tip) of the texture class. If FALSE no arrows are drawn.
<code>arrows.lty</code>	Single numerical. Line type of the arrows drawn outside the triangle, along each axis. Same possible types as <code>par("lty")</code> .
<code>points.type</code>	Single text letter. Point type. Either "p" (points only), "l" (lines only) or "b" (both points and lines), as for <code>plot()</code> or <code>points()</code> . Point refer here to soil texture values plotted on the triangle.
<code>pch</code>	Single numerical or vector of numerals, 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.
<code>z.type</code>	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()`.

<code>z.col.hue</code>	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).
<code>z.cex.range</code>	Vector of 2 numericals. Minimum and maximum 'cex' of the bubbles plotted on the triangle if 'z.name' is not NULL.
<code>z.pch</code>	Single numerical or vector of numericals. Point symbol number(s) to be used for the bubbles if 'z.name' is not NULL.
<code>text.tol</code>	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 $\geq \text{text.sum} * (1 - \text{text.tol})$ OR $\leq \text{text.sum} * (1 + \text{text.tol})$ . See 'text.sum' for more details, as well as 'tri.sum.tst' (to prevent texture sum tests).
<code>tri.sum.tst</code>	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.
<code>tri.pos.tst</code>	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).
<code>b.lim</code>	Vector of 2 numerical values. This is an equivalent to <code>plot()</code> <code>xlim</code> argument. Minimum and maximum x / bottom value of the texture triangle area, in FRACTION OF THE MAXIMAL EXTENSION. Default is <code>c(0,1)</code> . The real span is then <code>b.lim * text.sum</code> . This is a minimal 'zoom' implementation (results are not always perfect). 'b.lim' and 'l.lim' should be equal for better rendering.
<code>l.lim</code>	Vector of 2 numerical values. This is an equivalent to <code>plot()</code> <code>ylim</code> argument. Minimum and maximum y / left value of the texture triangle area, in FRACTION OF THE MAXIMAL EXTENSION. Default is <code>c(0,1)</code> . The real span is then <code>l.lim * text.sum</code> . 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  
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 [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**

<code>tri.data</code>	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).
<code>class.sys</code>	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).
<code>PiC.type</code>	Single character string. If equal to 'n', then a table of 0, 1, 2 or 3 is outputted (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 guaranteed for samples lying very close to an edge, or right on it. See < <a href="http://www.mail-archive.com/r-help@r-project.org/msg96180.html">http://www.mail-archive.com/r-help@r-project.org/msg96180.html</a> >
<code>css.names</code>	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.
<code>text.sum</code>	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 $\geq \text{text.sum} * (1 - \text{text.tol})$ OR $\leq \text{text.sum} * (1 + \text{text.tol})$ , 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.
<code>base.css.ps.lim</code>	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) <code>css.transf == TRUE</code> (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.
<code>tri.css.ps.lim</code>	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  $\geq \text{text.sum} * (1 - \text{text.tol})$  OR  $\leq \text{text.sum} * (1 + \text{text.tol})$ . 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 texture 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 [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" )

# 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   = "1"
) #

# 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.polygon.area

*Internal. 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 [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.polygon.centroids *Internal. 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 once 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](http://local.wasp.uwa.edu.au/~pbourke/geometry/polyarea/Calculating%20The%20Area%20And%20Centroid%20Of%20A%20Polygon). 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 [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.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 default parameters for all the current R session. 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 backup 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	<i>Internal. Stretch or reshape the range of value of some data set.</i>
--------	--

---

**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 [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.switch	<i>Internal. Used in the plot axis drawings.</i>
-----------	--

---

**Description**

Used in the plot axis drawings.

**Usage**

```
TT.switch(blr.clock = TT.get("blr.clock"), c1 = NA,
```

```
      c2 = NA, c3 = NA, c4 = NA, blr.order = c(1, 3,
```

```
      2))
```

**Arguments**

blr.clock  
 c1  
 c2  
 c3  
 c4  
 blr.order

**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	<i>Plot text labels for each values of a soil texture data table on an existing texture plot.</i>
---------	---

---

**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  
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 [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	<i>Log-linear transformation of a soil texture data table between 2 particle size systems (3 classes).</i>
----------------	--

---

**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  
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 [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 particle size classes in the inputted and outputted 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  
base.ps.lim  
dat.ps.lim  
text.sum  
text.tol  
tri.sum.tst  
tri.pos.tst
```

**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.ticks	<i>Internal. Plot the axis' ticks of a texture triangle plot.</i>
----------	---

---

**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.s = NULL, tri.sum.tst = NULL,  
  
         tri.pos.tst = FALSE, lwd.axis = NULL, col.axis = NULL)
```

**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 [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.ticks.lab	<i>Internal. Plot the axis ticks' labels of a texture triangle plot.</i>
--------------	--

---

**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, tlr.an = NULL, tk.ls = NULL,  
  
            tri.sum.tst = NULL, tri.pos.tst = FALSE, col.axis = NULL,  
  
            font.axis = NULL, cex.axis = NULL, family.op = 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 [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      *Internal. 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
- class.sys
- fg
- col
- cex
- font
- family.op
- adj
- pos
- offset
- blr.tx
- text.sum
- blr.clock

### 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.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

**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.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 [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.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, tri.pos.tst = NULL, set.par = FALSE,  
  
          blr.clock = NULL, text.sum = NULL)
```

**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 [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]

# Index

- \* **package**
  - soiltexture-package, 3
- data.frame, 10
- installed.packages, 7
- md5sum, 7
- package\_dependencies, 7
- read.table, 10
- select.list, 10
- soiltexture (soiltexture-package), 3
- soiltexture-package, 3
- soiltexture\_gui, 8
- soiltextureInfo, 6
- Sys.info, 7
- Sys.time, 7
- TT.add, 11
- TT.auto.set, 12
- TT.axis.arrows, 12
- TT.baseplot, 14
- TT.blr.ps.lim, 15
- TT.blr.tx.check, 16
- TT.check.ps.lim, 16
- TT.chemometrics.alr, 17
- TT.classes, 18
- TT.classes.tbl, 19
- TT.col2hsv, 20
- TT.contour, 20
- TT.css2xy, 22
- TT.data.test, 23
- TT.data.test.X, 24
- TT.dataset, 25
- TT.deg2rad, 25
- TT.dia2phi, 26
- TT.DJ.col, 26
- TT.edges, 27
- TT.env, 28
- TT.gen.op.set, 28
- TT.geo.get, 29
- TT.geo.set, 29
- TT.get, 30
- TT.grid, 31
- TT.ifelse, 32
- TT.image, 32
- TT.iwd, 34
- TT.kde2d, 35
- TT.lines, 36
- TT.locator, 37
- TT.mahalanobis, 38
- TT.normalise.sum, 40
- TT.normalise.sum.X, 40
- TT.par.op.set, 41
- TT.phi2dia, 42
- TT.plot, 10, 42
- TT.points, 58
- TT.points.in.classes, 60
- TT.polygon.area, 64
- TT.polygon.centroids, 65
- TT.set, 66
- TT.str, 67
- TT.switch, 67
- TT.text, 68
- TT.text.transf, 70
- TT.text.transf.X, 71
- TT.ticks, 72
- TT.ticks.lab, 73
- TT.vertices.plot, 74
- TT.vertices.tbl, 75
- TT.xy.grid, 75
- TT.xy2css, 76
- version, 7
- zpackages, 7