Package ‘Ternary’

December 3, 2021

Version 1.2.4
Title Create Ternary Plots
Description Plots ternary diagrams (simplex plots / Gibbs triangles) using the standard graphics functions.
An alternative to ‘ggtern’, which uses the ‘ggplot2’ family of plotting functions.
Includes a 'Shiny' user interface for point-and-click plotting.

URL https://ms609.github.io/Ternary/,
      https://github.com/ms609/Ternary/

BugReports https://github.com/ms609/Ternary/issues/
License GPL (>= 2)
Language en-GB
Depends R (>= 3.2.0)
Imports shiny, viridisLite,
Suggests colourpicker, knitr, readxl, rmarkdown, shinyjs, testthat (>= 3.0), vdiffr,

Config/Needs/coverage covr
Config/Needs/memcheck devtools, rcmdcheck
Config/Needs/metadata codemeta
Config/Needs/revdeps revdepcheck
Config/Needs/website pkgdown
Config/testthat/parallel false
Config/testthat/edition 3
LazyData true
VignetteBuilder knitr
Encoding UTF-8
RoxygenNote 7.1.2
NeedsCompilation no
Description

Plot shapes onto a ternary diagram created with `TernaryPlot()`.

Usage

```r
AddToTernary(PlottingFunction, coordinates, ...)

TernaryArrows(fromCoordinates, toCoordinates = fromCoordinates, ...)

TernaryLines(coordinates, ...)

TernaryPoints(coordinates, ...)

TernaryPolygon(coordinates, ...)
```
AddToTernary

TernaryText(coordinates, ...)

JoinTheDots(coordinates, ...)

Arguments

PlottingFunction
Function to add data to a plot; perhaps one of points, lines or text.

coordinates
A list, matrix, data.frame or vector in which each element (or row) specifies the three coordinates of a point in ternary space.

Additional parameters to pass to PlottingFunction(). If using TernaryText(), this will likely include the parameter labels, to specify the text to plot.

fromCoordinates, toCoordinates
For TernaryArrows(), coordinates at which arrows should begin and end; cf. x0, y0, x1 and y1 in arrows. Recycled as necessary.

Functions

• TernaryArrows: Add arrows
• TernaryLines: Add lines
• TernaryPoints: Add points
• TernaryPolygon: Add polygons
• TernaryText: Add text
• JoinTheDots: Add points, joined by lines

Author(s)

Martin R. Smith (martin.smith@durham.ac.uk)

Examples

coords <- list(
  A = c(1, 0, 2),
  B = c(1, 1, 1),
  C = c(1.5, 1.5, 0),
  D = c(0.5, 1.5, 1)
)
TernaryPlot()
AddToTernary(lines, coords, col='darkgreen', lty='dotted', lwd=3)
TernaryLines(coords, col='darkgreen')
TernaryArrows(coords[1], coords[2:4], col='orange', length=0.2, lwd=1)
TernaryText(coords, cex=0.8, col='red', font=2)
TernaryPoints(coords, pch=1, cex=2, col='blue')
AddToTernary(points, coords, pch=1, cex=3)
cbPalettes  

*Palettes compatible with colour blindness*

**Description**

Colour palettes recommended for use with colour blind audiences.

**Usage**

- cbPalette8
- cbPalette13
- cbPalette15

**Format**

Character vectors of lengths 8, 13 and 15.

- An object of class character of length 8.
- An object of class character of length 13.
- An object of class character of length 15.

**Details**

cbPalette15 is a [Brewer palette](http://mkweb.bcgsc.ca/biovis2012/color-blindness-palette.png). Because colours 4 and 7 are difficult to distinguish from colours 13 and 3, respectively, in individuals with tritanopia, cbPalette13 omits these colours (i.e. cbPalette13 <-cbPalette15[-c(4,7)]).

**Source**

- cbPalette8: Wong B. 2011. Color blindness. Nat. Methods. 8:441. doi: [10.1038/nmeth.1618](http://dx.doi.org/10.1038/nmeth.1618)

**Examples**

```r
data('cbPalette8')
plot.new()
plot.window(xlim = c(1, 16), ylim = c(0, 3))
text(1:8 * 2, 3, 1:8, col = cbPalette8)
points(1:8 * 2, rep(2, 8), col = cbPalette8, pch = 15)

data('cbPalette15')
text(1:15, 1, col = cbPalette15)
text(c(4, 7), 1, '[ ]')
points(1:15, rep(0, 15), col = cbPalette15, pch = 15)
```
ColourTernary

Description

Colour a ternary plot according to the output of a function

Usage

ColourTernary(
  values,
  spectrum = viridisLite::viridis(256L, alpha = 0.6),
  resolution = sqrt(ncol(values)),
  direction = getOption("ternDirection", 1L)
)

ColorTernary(
  values,
  spectrum = viridisLite::viridis(256L, alpha = 0.6),
  resolution = sqrt(ncol(values)),
  direction = getOption("ternDirection", 1L)
)

Arguments

values Numeric matrix, possibly created using TernaryPointValues(), with four named rows: x, y, cartesian coordinates of each triangle centre; z, value associated with that coordinate; down, triangle direction: 0 = point upwards; 1 = point downwards.
spectrum Vector of colours to use as a spectrum, or NULL to use values['z',].
resolution The number of triangles whose base should lie on the longest axis of the triangle. Higher numbers will result in smaller subdivisions and smoother colour gradients, but at a computational cost.
direction (optional) Integer specifying the direction that the current ternary plot should point: 1, up; 2, right; 3, down; 4, left.

Author(s)

Martin R. Smith (martin.smith@durham.ac.uk)

See Also

Other contour plotting functions: TernaryContour(), TernaryDensityContour(), TernaryPointValues()
Other functions for colouring and shading: TernaryTiles()
Examples

TernaryPlot(alab = 'a', blab = 'b', clab = 'c')

FunctionToContour <- function (a, b, c) {
  a - c + (4 * a * b) + (27 * a * b * c)
}

values <- TernaryPointValues(FunctionToContour, resolution = 24L)
ColourTernary(values)
TernaryContour(FunctionToContour, resolution = 36L)

TernaryPlot()
values <- TernaryPointValues(rgb, resolution = 20)
ColourTernary(values, spectrum = NULL)

# Create a helper function to place white centrally:
rgbWhite <- function (r, g, b) {
  highest <- apply(rbind(r, g, b), 2L, max)
  rgb(r/highest, g/highest, b/highest)
}

TernaryPlot()
values <- TernaryPointValues(rgbWhite, resolution = 20)
ColourTernary(values, spectrum = NULL)

OutsidePlot

Is a point in the plotting area?

Description

Evaluate whether a given set of coordinates lie outwith the boundaries of a plotted ternary diagram.

Usage

OutsidePlot(x, y, tolerance = 0)

Arguments

x, y       Vectors of x and y coordinates of points.
tolerance  Consider points this close to the edge of the plot to be inside. Set to negative values to count points that are just outside the plot as inside, and to positive values to count points that are just inside the margins as outside. Maximum positive value: 1/3.
ReflectedEquivalents

Value
OutsidePlot() returns a logical vector specifying whether each pair of x and y coordinates corresponds to a point outside the plotted ternary diagram.

Author(s)
Martin R. Smith (martin.smith@durham.ac.uk)

See Also
Other plot limits: TernaryXRange()

Examples

TernaryPlot()
points(0.5, 0.5, col = 'darkgreen')
OutsidePlot(0.5, 0.5)

points(0.1, 0.5, col = 'red')
OutsidePlot(0.1, 0.5)

OutsidePlot(c(0.5, 0.1), 0.5)

ReflectedEquivalents  Reflected equivalents of points outside the ternary plot

Description
To avoid edge effects, it may be desirable to add the value of a point within a ternary plot with the value of its 'reflection' across the nearest axis or corner.

Usage
ReflectedEquivalents(x, y, direction = getOption("ternDirection", 1L))

Arguments
x, y  Vectors of x and y coordinates of points.
direction  (optional) Integer specifying the direction that the current ternary plot should point: 1, up; 2, right; 3, down; 4, left.

Value
ReflectedEquivalents() returns a list of the x, y coordinates of the points produced if the given point is reflected across each of the edges or corners.
See Also

Other coordinate translation functions: TernaryCoords(), TriangleCentres(), XYSToTernary()

Examples

TernaryPlot(axis.labels=FALSE, point=4)

xy <- cbind(
    TernaryCoords(0.9, 0.08, 0.02),
    TernaryCoords(0.15, 0.8, 0.05),
    TernaryCoords(0.05, 0.1, 0.85)
)
x <- xy[1, ]
y <- xy[2, ]

points(x, y, col='red', pch=1:3)
ref <- ReflectedEquivalents(x, y)
points(ref[[1]][1], ref[[1]][2], col='blue', pch=1)
points(ref[[2]][1], ref[[2]][2], col='green', pch=2)
points(ref[[3]][1], ref[[3]][2], col='orange', pch=3)

---

TernaryApp

Graphical user interface for creating ternary plots

Description

TernaryApp() launches a 'Shiny' application for the construction of ternary plots. The 'app' allows data to be loaded and plotted, and provides code to reproduce the plot in R should more sophisticated plotting functions be desired.

Usage

TernaryApp()

Details

Load data:
The 'Load data' input tab allows for the upload of datasets. Data can be read from csv files, .txt files created with write.table(), or (if the 'readxl' package is installed) Excel spreadsheets. Data should be provided as three columns, corresponding to the three axes of the ternary plot. Colours or point styles may be specified in columns four to six to allow different categories of point to be plotted distinctly. Example datasets are installed at C:/Research/R/Ternary/inst/TernaryApp. Axes are automatically labelled using column names, if present; these can be edited manually on this tab.

Plot display:
Allows the orientation, colour and configuration of the plot and its axes to be adjusted,
**TernaryContour**

**Grids:**
Adjust the number, spacing and styling of major and minor grid lines.

**Labels:**
Configure the colour, position and size of tip and axis labels.

**Points:**
Choose whether to plot points, lines, connected points, or text. Set the style of points and lines.

**Exporting plots**
A plot can be saved to PDF or as a PNG bitmap at a specified size. Alternatively, R script that will generate the displayed plot can be viewed (using the ‘R code’ output tab) or downloaded to file.

**Author(s)**
Martin R. Smith (martin.smith@durham.ac.uk)

**References**
If you use figures produced with this package in a publication, please cite

**See Also**
Full detail of plotting with 'Ternary', including features not (yet) implemented in the application, is provided in the accompanying vignette.

---

TernaryContour  
*Add contours to a ternary plot*

**Description**
Draws contour lines to depict the value of a function in ternary space.

**Usage**

```r
TernaryContour(  
  Func,  
  resolution = 96L,  
  direction = getOption("ternDirection", 1L),  
  ...  
)
```
Arguments

- **Func**
  Function taking vectors of coordinates a, b and c, which returns a numeric vector whose value at each coordinate will be depicted.

- **resolution**
  The number of triangles whose base should lie on the longest axis of the triangle. Higher numbers will result in smaller subdivisions and smoother colour gradients, but at a computational cost.

- **direction**
  (optional) Integer specifying the direction that the current ternary plot should point: 1, up; 2, right; 3, down; 4, left.

- ... Further parameters to pass to `contour`.

Author(s)

Martin R. Smith (martin.smith@durham.ac.uk)

See Also

Other contour plotting functions: `ColourTernary()`, `TernaryDensityContour()`, `TernaryPointValues()`

Examples

```r
TernaryPlot(alab = 'a', blab = 'b', clab = 'c')

FunctionToContour <- function (a, b, c) {
  a - c + (4 * a * b) + (27 * a * b * c)
}

values <- TernaryPointValues(FunctionToContour, resolution = 24L)
ColourTernary(values)
TernaryContour(FunctionToContour, resolution = 36L)

# Note that FunctionToContour is sent a vector.
# Instead of
BadMax <- function (a, b, c) {
  max(a, b, c)
}

# Use
GoodMax <- function (a, b, c) {
  pmax(a, b, c)
}

TernaryPlot(alab = 'a', blab = 'b', clab = 'c')
ColourTernary(TernaryPointValues(GoodMax))
TernaryContour(GoodMax)

# Or, for a generalizable example,
GeneralMax <- function (a, b, c) {
  apply(rbind(a, b, c), 2, max)
}

TernaryPlot(alab = 'a', blab = 'b', clab = 'c')
ColourTernary(TernaryPointValues(GeneralMax))
```
TernaryCoords

Convert ternary coordinates to Cartesian space

Description

Convert coordinates of a point in ternary space, in the format \((a, b, c)\), to \(x\) and \(y\) coordinates of Cartesian space, which can be sent to standard functions in the 'graphics' package.

Usage

\[
\text{TernaryCoords(}
\begin{align*}
\text{abc,} \\
\text{b Coord = NULL,} \\
\text{c Coord = NULL,} \\
\text{direction = getOption(}\text{"ternDirection"}, \text{1L})
\end{align*}
\]

\#

\#

\text{S3 method for class 'matrix'}

\text{TernaryToXY(}
\begin{align*}
\text{abc,} \\
\text{b Coord = NULL,} \\
\text{c Coord = NULL,} \\
\text{direction = getOption(}\text{"ternDirection"}, \text{1L})
\end{align*}
\]

\#

\#

\text{S3 method for class 'numeric'}

\text{TernaryToXY(}
\begin{align*}
\text{abc,} \\
\text{b Coord = NULL,} \\
\text{c Coord = NULL,} \\
\text{direction = getOption(}\text{"ternDirection"}, \text{1L})
\end{align*}
\]

\text{Arguments}

\begin{itemize}
\item \text{abc} \hspace{1cm} \text{A vector of length three giving the position on a ternary plot that points in the direction specified by direction (1 = up, 2 = right, 3 = down, 4 = left).}
\end{itemize}
TernaryDensityContour

\[ \text{c}(100,0,0) \] will plot in the direction-most corner; \[ \text{c}(0,100,0) \] will plot in the corner clockwise of direction; \[ \text{c}(0,0,100) \] will plot in the corner anti-clockwise of direction. Alternatively, the a coordinate can be specified as the first parameter, in which case the b and c coordinates must be specified via \text{b}_\text{coord} \text{ and } \text{c}_\text{coord}. Or, a matrix with three rows, representing in turn the a, b and c coordinates of points.

- \text{b}_\text{coord} \quad \text{The b coordinate, if abc is a single number.}
- \text{c}_\text{coord} \quad \text{The c coordinate, if abc is a single number.}
- \text{direction} \quad \text{(optional) Integer specifying the direction that the current ternary plot should point: } 1, \text{ up; } 2, \text{ right; } 3, \text{ down; } 4, \text{ left.}

**Value**

\text{TernaryCoords()} \text{ returns a vector of length two that converts the coordinates given in abc into Cartesian } (x, y) \text{ coordinates corresponding to the plot created by the last call of } \text{TernaryPlot()}.\]

**Author(s)**

Martin R. Smith (martin.smith@durham.ac.uk)

**See Also**

- \text{TernaryPlot()}

Other coordinate translation functions: \text{ReflectedEquivalents()}, \text{TriangleCentres()}, \text{XYToTernary()}

**Examples**

\text{TernaryCoords(100, 0, 0)}
\text{TernaryCoords(c(0, 100, 0))}

\text{coords <- matrix(1:12, nrow = 3)}
\text{TernaryToXY(coords)}
Usage

TernaryDensityContour(

coordinates,

bandwidth,

resolution = 25L,

tolerance = -0.2/resolution,

direction = getOption("ternDirection", 1L),

...
)

Arguments

coordinates  A list, matrix, data.frame or vector in which each element (or row) specifies the three coordinates of a point in ternary space.

bandwidth  Vector of bandwidths for x and y directions. Defaults to normal reference bandwidth (see MASS::bandwidth.nrd). A scalar value will be taken to apply to both directions.

resolution  The number of triangles whose base should lie on the longest axis of the triangle. Higher numbers will result in smaller subdivisions and smoother colour gradients, but at a computational cost.

tolerance  Numeric specifying how close to the margins the contours should be plotted, as a fraction of the size of the triangle. Negative values will cause contour lines to extend beyond the margins of the plot.

direction  (optional) Integer specifying the direction that the current ternary plot should point: 1, up; 2, right; 3, down; 4, left.

...  Further parameters to pass to `contour`.

Details

This function is modelled on MASS::kde2d(), which uses "an axis-aligned bivariate normal kernel, evaluated on a square grid".

This is to say, values are calculated on a square grid, and contours fitted between these points. This produces a couple of artefacts. Firstly, contours may not extend beyond the outermost point within the diagram, which may fall some distance from the margin of the plot if a low resolution is used. Setting a negative tolerance parameter allows these contours to extend closer to (or beyond) the margin of the plot.

Individual points cannot fall outside the margins of the ternary diagram, but their associated kernels can. In order to sample regions of the kernels that have 'bled' outside the ternary diagram, each point’s value is calculated by summing the point density at that point and at equivalent points outside the ternary diagram, 'reflected' across the margin of the plot (see function ReflectedEquivalents). This correction can be disabled by setting the edgeCorrection parameter to FALSE.

A model based on a triangular grid may be more appropriate in certain situations, but is non-trivial to implement; if this distinction is important to you, please let the maintainers known by opening a Github issue.
Author(s)

Adapted from MASS::kde2d() by Martin R. Smith

See Also

Other contour plotting functions: ColourTernary(), TernaryContour(), TernaryPointValues()

Examples

TernaryPlot(axis.labels = seq(0, 10, by = 1))

nPoints <- 400L
coordinates <- cbind(abs(rnorm(nPoints, 2, 3)),
                    abs(rnorm(nPoints, 1, 1.5)),
                    abs(rnorm(nPoints, 1, 0.5)))

ColourTernary(TernaryDensity(coordinates, resolution = 10L))
TernaryPoints(coordinates, col = 'red', pch = '.')
TernaryDensityContour(coordinates, resolution = 30L)

---

TernaryPlot

Create a ternary plot

Description

Create and style a blank ternary plot.

Usage

TernaryPlot(
  atip = NULL,
  btip = NULL,
  ctip = NULL,
  alab = NULL,
  blab = NULL,
  clab = NULL,
  lab.offset = 0.16,
  lab.col = NULL,
  point = "up",
  clockwise = TRUE,
  xlim = NULL,
  ylim = NULL,
  lab.cex = 1,
  lab.font = 0,
  tip.cex = lab.cex,
  tip.font = 2,
Arguments

atip, btip, ctip

Character string specifying text to title corners, proceeding clockwise from the corner specified in point (default: top).

alab, blab, clab

Character string specifying text with which to label the corresponding sides of the triangle. Left or right-pointing arrows are produced by typing \U2190 or
lab.offset Numeric specifying distance between midpoint of axis label and the axis. Increase padding if labels are being clipped. Use a vector of length three to specify a different offset for each label.

lab.col Character vector specifying colours for axis labels. Use a vector of length three to specify a different colour for each label.

point Character string specifying the orientation of the ternary plot: should the triangle point "up", "right", "down" or "left"? The integers 1 to 4 can be used in place of the character strings.

clockwise Logical specifying the direction of axes. If TRUE (the default), each axis runs from zero to its maximum value in a clockwise direction around the plot.

xlim, ylim Numeric vectors of length 2 specifying the minimum and maximum x and y limits of the plotted area, to which padding will be added. The default is to display the complete height or width of the plot. Allows cropping to magnified region of the plot. (See vignette for diagram.) May be overridden if isometric=TRUE; see documentation of isometric parameter.

lab.cex, tip.cex Numeric specifying character expansion (font size) for axis labels. Use a vector of length three to specify a different value for each direction.

lab.font, tip.font Numeric specifying font style (Roman, bold, italic, bold-italic) for axis titles. Use a vector of length three to set a different font for each direction.

isometric Logical specifying whether to enforce an equilateral shape for the ternary plot. If only one of xlim and ylim is set, the other will be calculated to maintain an equilateral plot. If both xlim and ylim are set, but have different ranges, then the limit with the smaller range will be scaled until its range matches that of the other limit.

atip.rotate, btip.rotate, ctip.rotate Integer specifying number of degrees to rotate label of rightmost apex.

atip.pos, btip.pos, ctip.pos Integer specifying positioning of labels, iff the corresponding xlab.rotate parameter is set.

padding Numeric specifying size of internal margin of the plot; increase if axis labels are being clipped.

col The colour for filling the plot; see polygon.

grid.lines Integer specifying the number of grid lines to plot.

grid.col, grid.minor.col Colours to draw the grid lines. Use a vector of length three to set different values for each direction.

grid.lty, grid.minor.lty Character or integer vector; line type of the grid lines. Use a vector of length three to set different values for each direction.

grid.lwd, grid.minor.lwd Non-negative numeric giving line width of the grid lines. Use a vector of length three to set different values for each direction.
grid.minor.lines  Integer specifying the number of minor (unlabelled) grid lines to plot between each major pair.

axis.lty  Line type for both the axis line and tick marks. Use a vector of length three to set a different value for each direction.

axis.labels  This can either be a logical value specifying whether (numerical) annotations are to be made at the tickmarks, or a character or expression vector of labels to be placed at the tick points.

axis.cex  Numeric specifying character expansion (font size) for axis labels. Use a vector of length three to set a different value for each direction.

axis.font  Font for text. Defaults to \texttt{par('font')}.

axis.rotate  Logical specifying whether to rotate axis labels to parallel grid lines, or numeric specifying custom rotation for each axis, to be passed as \texttt{srt} parameter to \texttt{text()}. Expand margins or set \texttt{par(xpd = NA)} if labels are clipped.

axis.pos  Vector of length one or three specifying position of axis labels, to be passed as \texttt{pos} parameter to \texttt{text()}; populated automatically if \texttt{NULL} (the default).

axis.tick  Logical specifying whether to mark the axes with tick marks.

axis.lwd, ticks.lwd  Line width for the axis line and tick marks. Zero or negative values will suppress the line or ticks. Use a vector of length three to set different values for each axis.

ticks.length  Numeric specifying distance that ticks should extend beyond the plot margin. Also affects position of axis labels, which are plotted at the end of each tick. Use a vector of length three to set a different length for each direction.

axis.col, ticks.col, tip.col  Colours for the axis line, tick marks and tip labels respectively. Use a vector of length three to set a different value for each direction. \texttt{axis.col = NULL} means to use \texttt{par('fg')}, possibly specified inline, and \texttt{ticks.col = NULL} means to use whatever colour \texttt{axis.col} resolved to.

...  Additional parameters to \texttt{plot}.

direction  (optional) Integer specifying the direction that the current ternary plot should point: 1, up; 2, right; 3, down; 4, left.

Details

The plot will be generated using the standard 'graphics' plot functions, on which additional elements can be added using cartesian coordinates, perhaps using functions such as \texttt{arrows}, \texttt{legend} or \texttt{text}.

Functions

- \texttt{HorizontalGrid}: Add \texttt{grid.lines} horizontal lines to the ternary plot

Author(s)

Martin R. Smith (martin.smith@durham.ac.uk)
TernaryPointValues

See Also

- `AddToTernary()`: Add elements to a ternary plot
- `TernaryCoords()`: Convert ternary coordinates to Cartesian \( (x, y) \) coordinates
- `TernaryXRange(), TernaryYRange()`: What are the \( x \) and \( y \) limits of the plotted region?

Examples

```r
TernaryPlot(atip = "Top", btip = "Bottom", ctip = "Right", axis.col = "red",
col = rgb(0.8, 0.8, 0.8))
HorizontalGrid(grid.lines = 2, grid.col = 'blue', grid.lty = 1)
# the second line corresponds to the base of the triangle, and is not drawn
```

TernaryPointValues

Value of a function at regularly spaced points

Description

Intended to facilitate coloured contour plots with `ColourTernary()`. `TernaryPointValue()` evaluates a function at points on a triangular grid; `TernaryDensity()` calculates the density of points in each grid cell.

Usage

```r
TernaryPointValues(
  Func,
  resolution = 48L,
  direction = getOption("ternDirection", 1L),
  ...
)
```

```r
TernaryDensity(
  coordinates,
  resolution = 48L,
  direction = getOption("ternDirection", 1L)
)
```

Arguments

- `Func`: Function taking vectors of coordinates \( a, b \) and \( c \), which returns a numeric vector whose value at each coordinate will be depicted.
- `resolution`: The number of triangles whose base should lie on the longest axis of the triangle. Higher numbers will result in smaller subdivisions and smoother colour gradients, but at a computational cost.
- `direction`: (optional) Integer specifying the direction that the current ternary plot should point: 1, up; 2, right; 3, down; 4, left.
- `...`: Additional parameters to `Func()`.
- `coordinates`: A list, matrix, data.frame or vector in which each element (or row) specifies the three coordinates of a point in ternary space.
TernaryTiles

Value

TernaryPointValues() returns a matrix whose rows correspond to:

- \( x, y \): co-ordinates of the centres of smaller triangles
- \( z \): The value of \( \text{Func}(a,b,c) \), where \( a, b \) and \( c \) are the ternary coordinates of \( x \) and \( y \).
- \( \text{down} \): 0 if the triangle concerned points upwards (or right), 1 otherwise

Author(s)

Martin R. Smith (martin.smith@durham.ac.uk)

See Also

Other contour plotting functions: ColourTernary(), TernaryContour(), TernaryDensityContour()

Examples

TernaryPointValues(function (a, b, c) a * b * c, resolution = 2)

TernaryPlot(grid.lines = 4)
cols <- TernaryPointValues(rgb, resolution = 4)
text(as.numeric(cols['x',]), as.numeric(cols['y',]),
     labels = ifelse(cols['down',] == '1', 'v', '^'),
     col = cols['z',])

TernaryPlot(axis.labels = seq(0, 10, by = 1))

nPoints <- 4000L
coordinates <- cbind(abs(rnorm(nPoints, 2, 3)),
                     abs(rnorm(nPoints, 1, 1.5)),
                     abs(rnorm(nPoints, 1, 0.5)))
density <- TernaryDensity(coordinates, resolution = 10L)
ColourTernary(density)
TernaryPoints(coordinates, col = 'red', pch = '.')

TernaryTiles

Paint tiles on ternary plot

Description

Function to fill a ternary plot with coloured tiles. Useful in combination with TernaryPointValues and TernaryContour.
Usage

TernaryTiles(
  x,
  y,
  down,
  resolution,
  col,
  direction = getOption("ternDirection", 1L)
)

Arguments

x, y          Numeric vectors specifying x and y coordinates of centres of each triangle.
down         Logical vector specifying TRUE if each triangle should point down (or right), FALSE otherwise.
resolution   The number of triangles whose base should lie on the longest axis of the triangle. Higher numbers will result in smaller subdivisions and smoother colour gradients, but at a computational cost.
col          Vector specifying the colour with which to fill each triangle.
direction    (optional) Integer specifying the direction that the current ternary plot should point: 1; up; 2; right; 3, down; 4, left.

Author(s)

Martin R. Smith (martin.smith@durham.ac.uk)

See Also

Other functions for colouring and shading: ColourTernary()

Examples

FunctionToContour <- function (a, b, c) {
  a - c + (4 * a * b) + (27 * a * b * c)
}

TernaryPlot()

values <- TernaryPointValues(FunctionToContour, resolution = 24L)
ColourTernary(values)
TernaryContour(FunctionToContour, resolution=36L)
TernaryXRange

Description

X and Y coordinates of ternary plotting area

Usage

TernaryXRange(direction = getOption("ternDirection", 1L))

TernaryYRange(direction = getOption("ternDirection", 1L))

Arguments

direction (optional) Integer specifying the direction that the current ternary plot should point: 1, up; 2, right; 3, down; 4, left.

Value

TernaryXRange() and TernaryYRange() return the minimum and maximum X or Y coordinate of the area in which a ternary plot is drawn, oriented in the specified direction. Because the plotting area is a square, the triangle of the ternary plot will not occupy the full range in one direction. Assumes that the defaults have not been overwritten by specifying xlim or ylim.

Functions

- TernaryYRange: Returns the minimum and maximum Y coordinate for a ternary plot in the specified direction.

Author(s)

Martin R. Smith (martin.smith@durham.ac.uk)

See Also

Other plot limits: OutsidePlot()
TriangleCentres

Coordinates of triangle mid-points

Description

Calculate x and y coordinates of the midpoints of triangles tiled to cover a ternary plot.

Usage

TriangleCentres(resolution = 48L, direction = getOption("ternDirection", 1L))

Arguments

resolution The number of triangles whose base should lie on the longest axis of the triangle. Higher numbers will result in smaller subdivisions and smoother colour gradients, but at a computational cost.
direction (optional) Integer specifying the direction that the current ternary plot should point: 1, up; 2, right; 3, down; 4, left.

Value

TriangleCentres() returns a matrix with three named rows:

• x x coordinates of triangle midpoints;
• y y coordinates of triangle midpoints;
• triDown 0 for upwards-pointing triangles, 1 for downwards-pointing.

Author(s)

Martin R. Smith (martin.smith@durham.ac.uk)

See Also

Other coordinate translation functions: ReflectedEquivalents(), TernaryCoords(), XYToTernary()

Examples

TernaryPlot(grid.lines = 4)
centres <- TriangleCentres(4)
text(centres["x", ], centres["y", ], ifelse(centres["triDown", ], "v", "^")))
Description

Convert cartesian \((x, y)\) coordinates to a point in ternary space.

Usage

\[
\text{XYToTernary}(x, y, \text{direction} = \text{getOption("ternDirection", 1L)})
\]

Arguments

- \(x, y\) Numeric values giving the \(x\) and \(y\) coordinates of a point or points.
- \text{direction} (optional) Integer specifying the direction that the current ternary plot should point: 1, up; 2, right; 3, down; 4, left.

Value

XYToTernary() Returns the ternary point(s) corresponding to the specified \(x\) and \(y\) coordinates, where \(a + b + c = 1\).

Author(s)

Martin R. Smith (martin.smith@durham.ac.uk)

See Also

Other coordinate translation functions: \text{ReflectedEquivalents()}, \text{TernaryCoords()}, \text{TriangleCentres}()

Examples

XYToTernary(c(0.1, 0.2), 0.5)
Index

* contour plotting functions
  ColourTernary, 5
  TernaryContour, 9
  TernaryDensityContour, 12
  TernaryPointValues, 18
* coordinate translation functions
  ReflectedEquivalents, 7
  TernaryCoords, 11
  TriangleCentres, 22
  XYToTernary, 23
* datasets
  cbPalettes, 4
* functions for colouring and shading
  ColourTernary, 5
  TernaryTiles, 19
* plot limits
  OutsidePlot, 6
  TernaryXRange, 21

AddToTernary, 2
AddToTernary(), 18
arrows, 3, 17

cbPalette13 (cbPalettes), 4
cbPalette15 (cbPalettes), 4
cbPalette8 (cbPalettes), 4
cbPalettes, 4
ColorTernary (ColourTernary), 5
ColourTernary, 5, 10, 14, 19, 20
ColourTernary(), 18
contour, 10, 13

HorizontalGrid (TernaryPlot), 14
JoinTheDots (AddToTernary), 2

legend, 17
lines, 3

OutsidePlot, 6, 21