

# Package ‘Ternary’

April 15, 2019

**Version** 1.0.2.9000

**Date** 2018-12-10

**Title** An R Package for Creating Ternary Plots

**Description** Plots ternary diagrams using the standard graphics functions.

An alternative to 'ggtern', which uses the 'ggplot2' family of plotting functions.

**URL** <https://ms609.github.io/Ternary>

**BugReports** <https://github.com/ms609/Ternary/issues>

**License** GPL (>= 2)

**Language** en-GB

**Depends** R (>= 3.2.0)

**Suggests** knitr,  
rmarkdown,  
testthat

**LazyData** true

**VignetteBuilder** knitr

**Encoding** UTF-8

**RoxygenNote** 6.1.1

**Roxygen** list(markdown = TRUE)

## R topics documented:

AddToTernary . . . . .	2
cbPalette15 . . . . .	3
cbPalette8 . . . . .	4
TernaryCoords . . . . .	4
TernaryPlot . . . . .	5
TernaryXRange . . . . .	8

<b>Index</b>	<b>9</b>
--------------	----------

---

`AddToTernary`*Add elements to ternary plot*

---

### Description

Plot points onto a ternary diagram created with [TernaryPlot](#).

### Usage

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

```
TernaryPoints(coordinates, ...)
```

```
TernaryText(coordinates, ...)
```

```
TernaryLines(coordinates, ...)
```

```
TernaryPolygon(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.

### Functions

- `TernaryPoints`: Add points
- `TernaryText`: Add points
- `TernaryLines`: Add points
- `TernaryPolygon`: Add points
- `JoinTheDots`: Add points, joined by lines

### Author(s)

Martin R. Smith

**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='green', lwd=2)
  TernaryLines(coords, col='red', lty='dotted')
  TernaryText(coords, cex=0.7, col='red')
  TernaryPoints(coords, pch=1, cex=2, col='blue')
  AddToTernary(points, coords, pch=1, cex=3)
}
```

---

cbPalette15

*Fifteen-colour palette compatible with colour blindness*


---

**Description**

A fifteen-colour **Brewer palette** comprehensible by colour blind viewers.

**Usage**

```
cbPalette15
```

**Format**

An object of class character of length 15.

**Details**

Note that colour 4 is difficult to distinguish from colour 13 in individuals with tritanopia. Likewise, colour 7 is difficult to distinguish from colour 3. You may wish to use `cbPalette13 <-cbPalette15[-c(4,7)]`.

**Source**

<http://mkweb.bcgsc.ca/biovis2012/color-blindness-palette.png>

**See Also**

[cbPalette8](#)

---

cbPalette8	<i>Eight-colour palette compatible with colour blindness</i>
------------	--------------------------------------------------------------

---

**Description**

An eight-colour palette recommended for use with colour blind audiences.

**Usage**

```
cbPalette8
```

**Format**

An object of class character of length 8.

**Source**

Wong B. 2011. Color blindness. *Nat. Methods*. 8:441. doi: [10.1038/nmeth.1618](https://doi.org/10.1038/nmeth.1618)

**See Also**

[cbPalette15](#)

---

TernaryCoords	<i>Convert ternary coordinates to Cartesian space</i>
---------------	-------------------------------------------------------

---

**Description**

Converts 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**

```
TernaryCoords(abc, b_coord = NULL, c_coord = NULL,
  direction = getOption("ternDirection"))
```

**Arguments**

abc	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). $c(100, 0, 0)$ will plot in the direction-most corner; $c(0, 100, 0)$ will plot in the corner clockwise of direction; $c(0, 0, 100)$ 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 b_coord and c_coord.
b_coord	The b coordinate, if abc is a single number.

c_coord	The c coordinate, if abc is a single number.
direction	(optional) Integer specifying the direction that the current ternary plot should point: 1, up; 2, right; 3, down; 4, left.

**Value**

A vector of length two that converts the coordinates given in abc into Cartesian (x, y) coordinates corresponding to the plot created by the last call of [TernaryPlot](#).

**Author(s)**

Martin R. Smith

**See Also**

[TernaryPlot](#)

---

TernaryPlot	<i>Create a ternary plot</i>
-------------	------------------------------

---

**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, point = "up",
  xlim = NULL, ylim = NULL, lab.cex = 1, lab.font = 0,
  tip.cex = lab.cex, tip.font = 2, isometric = TRUE,
  atip.rotate = NULL, btip.rotate = NULL, ctip.rotate = NULL,
  atip.pos = NULL, btip.pos = NULL, ctip.pos = NULL,
  padding = 0.08, col = NA, grid.lines = 10, grid.col = "darkgrey",
  grid.lty = "solid", grid.lwd = par("lwd"), grid.minor.lines = 4,
  grid.minor.col = "lightgrey", grid.minor.lty = "solid",
  grid.minor.lwd = par("lwd"), axis.lty = "solid",
  axis.labels = TRUE, axis.cex = 0.8, axis.font = par("font"),
  axis.tick = TRUE, axis.lwd = 1, ticks.lwd = axis.lwd,
  ticks.length = 0.025, axis.col = "black", ticks.col = grid.col,
  axis.labels.col = axis.col, ...)
```

```
HorizontalGrid(grid.lines = 10, grid.col = "grey",
  grid.lty = "dotted", grid.lwd = par("lwd"),
  direction = getOption("ternDirection"))
```

**Arguments**

<code>atip, btip, ctip</code>	Character specifying text to title corners, proceeding clockwise from the corner specified in <code>point</code> (default: top).
<code>alab, blab, clab</code>	Character specifying text with which to label the corresponding sides of the triangle. Left or right-pointing arrows are produced by typing <code>\U2190</code> or <code>\U2192</code> , or using expression( <code>'value' %&gt;% ' '</code> ).
<code>lab.offset</code>	Numeric specifying distance between midpoint of axis label and the axis. Increase padding if labels are being clipped.
<code>point</code>	Character specifying the orientation of the ternary plot: should the triangle point up, left, right or down?
<code>xlim, ylim</code>	Numeric vectors of length 2 specifying the minimum and maximum $x$ and $y$ limits of the plotted area, to which padding will be added. Presently overrides the setting of <code>isometric</code> . Allows cropping to magnified region of the plot. (See vignette for diagram.)
<code>lab.cex, tip.cex</code>	Numeric specifying character expansion for axis titles.
<code>lab.font, tip.font</code>	Numeric specifying font (roman, bold, italic, bold-italic) for axis titles.
<code>isometric</code>	Logical specifying whether to enforce an equilateral shape for the ternary plot. Presently ignored if <code>xlim</code> or <code>ylim</code> are set.
<code>atip.rotate, btip.rotate, ctip.rotate</code>	Integer specifying number of degrees to rotate label of rightmost apex.
<code>atip.pos, btip.pos, ctip.pos</code>	Integer specifying positioning of labels, iff corresponding <code>xlab.rotate</code> parameter is set.
<code>padding</code>	Numeric specifying size of internal margin of the plot; increase if axis labels are being clipped.
<code>col</code>	The colour for filling the plot; see <code>[graphics:polygon]</code> .
<code>grid.lines</code>	Integer specifying the number of grid lines to plot.
<code>grid.col, grid.minor.col</code>	The colour to draw the grid lines.
<code>grid.lty, grid.minor.lty</code>	Character or (integer) numeric; line type of the grid lines.
<code>grid.lwd, grid.minor.lwd</code>	Non-negative numeric giving line width of the grid lines.
<code>grid.minor.lines</code>	Integer specifying the number of minor (unlabelled) grid lines to plot between each major pair.
<code>axis.lty</code>	Line type for both the axis line and tick marks
<code>axis.labels</code>	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.

<code>axis.cex</code>	Numeric specifying character expansion for axis labels.
<code>axis.font</code>	Font for text. Defaults to <code>par('font')</code> .
<code>axis.tick</code>	Logical specifying whether to mark the axes with tick marks.
<code>axis.lwd, ticks.lwd</code>	Line width for the axis line and tick marks. Zero or negative values will suppress the line or ticks.
<code>ticks.length</code>	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.
<code>axis.col, ticks.col, axis.labels.col</code>	Colours for the axis line, tick marks and labels, respectively. <code>axis.col = NULL</code> means to use <code>par('fg')</code> , possibly specified inline, and <code>ticks.col = NULL</code> means to use whatever colour <code>axis.col</code> resolved to.
<code>...</code>	Additional parameters to <code>[graphics:plot]</code> .
<code>direction</code>	(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 [arrows](#), [legend](#) or [text](#).

## Functions

- `HorizontalGrid`: Add `grid.lines` horizontal lines to the ternary plot

## Author(s)

Martin R. Smith

## See Also

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

## Examples

```
{
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
}
```

---

TernaryXRange	<i>X and Y coordinates of ternary plotting area</i>
---------------	-----------------------------------------------------

---

**Description**

X and Y coordinates of ternary plotting area

**Usage**

```
TernaryXRange(direction = getOption("ternDirection"))
```

```
TernaryYRange(direction = getOption("ternDirection"))
```

**Arguments**

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

**Value**

Returns the minimum and maximum X coordinate for a ternary plot, oriented in the specified direction.

**Functions**

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

**Author(s)**

Martin R. Smith



# Index

## \*Topic **datasets**

cbPalette15, [3](#)

cbPalette8, [4](#)

AddToTernary, [2](#), [7](#)

cbPalette15, [3](#), [4](#)

cbPalette8, [3](#), [4](#)

HorizontalGrid (TernaryPlot), [5](#)

JoinTheDots (AddToTernary), [2](#)

legend, [7](#)

lines, [2](#)

points, [2](#)

TernaryCoords, [4](#), [7](#)

TernaryLines (AddToTernary), [2](#)

TernaryPlot, [2](#), [5](#), [5](#)

TernaryPoints (AddToTernary), [2](#)

TernaryPolygon (AddToTernary), [2](#)

TernaryText (AddToTernary), [2](#)

TernaryXRange, [7](#), [8](#)

TernaryYRange, [7](#)

TernaryYRange (TernaryXRange), [8](#)

text, [2](#), [7](#)