Package ‘ggtern’

April 11, 2020

Version 3.3.0
Date 2020-04-9
Title An Extension to ‘ggplot2’, for the Creation of Ternary Diagrams
Description Extends the functionality of ‘ggplot2’, providing the capability to plot ternary diagrams for (subset of) the ‘ggplot2’ geometries. Additionally, ‘ggtern’ has implemented several NEW geometries which are unavailable to the standard ‘ggplot2’ release. For further examples and documentation, please proceed to the ‘ggtern’ website.

Author Nicholas Hamilton [aut, cre]
Maintainer Nicholas Hamilton <nick@ggtern.com>
Depends R (>= 2.10), ggplot2 (>= 3.3.0)
Imports compositions (>= 1.40-2), grid, gridExtra, gtable (>= 0.1.2), latex2exp (>= 0.4), MASS, plyr (>= 1.8.3), scales (>= 0.3.0), stats, proto, utils, lattice, hexbin

Enhances sp
License GPL-2 | file LICENSE
Encoding UTF-8
URL http://www.ggtern.com

R topics documented:

'theme-ticks.R' 'theme-showtitles.R' 'theme-ticksoutside.R'
'theme-zoom.R' 'utilities.R' 'utilities-help.R'
'geom-density-tern.R' 'stat-density-tern.R' 'geom-mask.R'
'geom-Xline.R' 'geom-Xisoprop.R' 'geom-confidence-tern.R'
'stat-confidence-tern.R' 'geom-errorbarX.R'
'geom-smooth-tern.R' 'stat-smooth-tern.R' 'geom-mean-ellipse.R'
'stat-mean-ellipse.R' 'geom-interpolate-tern.R'
'stat-interpolate-tern.R' 'stat-interpolate-methods.R'
'geom-crosshair-tern.R' 'geom-point-swap.R' 'geom-hex-tern.R'
'stat-hex-tern.R' 'annotation-tern.R'
'annotation-raster-tern.R' 'geom-text-viewport.R'
'geom-label-viewport.R' 'geom-polygon-closed.R'
'geom-tri-tern.R' 'stat-tri-tern.R'

NeedsCompilation no
Repository CRAN
RoxygenNote 6.1.1
Date/Publication 2020-04-11 14:50:05 UTC

R topics documented:

.getFunctions .................................................. 4
aes ................................................................. 4
annotate .......................................................... 5
annotation_raster_tern ........................................ 6
approved_layers ................................................. 7
arrangeGrob ...................................................... 9
breaks_tern ....................................................... 10
coord_tern ....................................................... 11
data_Feldspar .................................................. 12
data_Fragments ................................................ 13
data_SkyeLava .................................................. 14
data_USDA ......................................................... 15
data_WhiteCells ............................................... 16
draw_key_tern ................................................... 17
g geom_confidence_tern ........................................ 18
g geom_crosshair_tern ......................................... 20
g geom_density_tern ........................................... 22
g geom_errorbarX ............................................... 24
g geom_hex_tern ................................................ 27
g geom_interpolate_tern ....................................... 29
g geom_label_viewport ......................................... 31
g geom_mask ....................................................... 33
g geom_mean_ellipse .......................................... 34
g geom_point_swap ............................................. 36
g geom_polygon_closed ........................................ 37
g geom_smooth_tern ............................................ 38
R topics documented:

geom_text_viewport .................................................. 40
geom_tri_tern ............................................................. 42
geom_Xisoprop ............................................................ 44
geom_Xline ................................................................. 45
ggplot ................................................................. 47
ggsave ............................................................... 48
ggtern ................................................................. 49
ggtern_labels ............................................................ 50
ggtern_labels_arrow_suffix ............................................ 52
ggtern_package .......................................................... 53
ggtern_themes ............................................................ 55
labels_tern .............................................................. 58
label_formatter .......................................................... 59
mahalanobis_distance .................................................. 59
position_jitter_tern ..................................................... 60
position_nudge_tern ..................................................... 60
predictdf2d .............................................................. 61
scale_X_continuous ....................................................... 61
strip_unapproved ......................................................... 62
ternary_transformation .................................................. 63
tern_limits ............................................................... 64
theme_arrowlength ........................................................ 65
theme_bordersontop ....................................................... 67
theme_clockwise ........................................................... 68
theme_complete ........................................................... 68
theme_convenience_functions ......................................... 69
theme_elements ........................................................... 71
theme_gridsontop ......................................................... 73
theme_latex ............................................................... 74
theme_legend_position ................................................... 75
theme_mesh ............................................................... 75
theme_noarrows .......................................................... 76
theme_nomask ............................................................. 76
theme_novar_tern ........................................................ 77
theme_rotate ............................................................. 78
theme_showgrid .......................................................... 78
theme_showlabels ........................................................ 80
theme_showprimary ....................................................... 80
theme_showtitles ......................................................... 82
theme_ticklength ........................................................ 82
theme_ticksoutside ....................................................... 83
theme_zoom_X ............................................................ 83
zzz-depreciated .......................................................... 85

Index 88
### Description

OLD FUNCTIONS `new_panel`, `train_layout`, `train_position`, `train_ranges`, `map_position`, `map_layout`, `reset_scales`, `facet_render`, `xlabel`, `ylabel` expand_default`, ## REMOVED

### Usage

```
.getFunctions()
```

### aes

**Modified Aesthetic Mappings**

### Description

Modified Aesthetic Mappings

### Usage

```
aes(x, y, z, ...)
```

### Arguments

- **x**: x value
- **y**: y value
- **z**: z value
- **...**: other arguments as per `aes`

### Details

An extension to the base `aes` function from ggplot2, this is modified to handle a default z mapping for application in ternary phase diagrams. Does not alter the standard behaviour.

### See Also

Parent `aes` function.
annotate

Create an annotation layer (ggtern version).

**Description**

This function adds geoms to a plot. Unlike typical a geom function, the properties of the geoms are not mapped from variables of a data frame, but are instead passed in as vectors. This is useful for adding small annotations (such as text labels) or if you have your data in vectors, and for some reason don’t want to put them in a data frame.

**Usage**

```r
annotate(geom, x = NULL, y = NULL, z = NULL, xmin = NULL, xmax = NULL, ymin = NULL, ymax = NULL, zmin = NULL, zmax = NULL, xend = NULL, yend = NULL, zend = NULL, ..., na.rm = FALSE)
```

**Arguments**

- `geom` name of geom to use for annotation
- `x, y, z, xmin, ymin, zmin, xmax, ymax, zmax, xend, yend, zend` positioning aesthetics - you must specify at least one of these.
- `...` Other arguments passed on to `layer()`. These are often aesthetics, used to set an aesthetic to a fixed value, like `colour = "red"` or `size = 3`. They may also be parameters to the paired geom/stat.
- `na.rm` If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.

**Details**

Note that all position aesthetics are scaled (i.e. they will expand the limits of the plot so they are visible), but all other aesthetics are set. This means that layers created with this function will never affect the legend.

**Author(s)**

Nicholas Hamilton

**See Also**

annotate
Examples

```r
ggtern() +
annotate(geom = 'text',
    x = c(0.5,1/3,0.0),
    y = c(0.5,1/3,0.0),
    z = c(0.0,1/3,1.0),
    angle = c(0,30,60),
    vjust = c(1.5,0.5,-0.5),
    label = paste("Point",c("A","B","C")),
    color = c("green","red","blue")) +
theme_dark() +
theme_nomask()
```

Description

This is a special version of `geom_raster` optimised for static annotations that are the same in every panel. These annotations will not affect scales (i.e. the x and y axes will not grow to cover the range of the raster, and the raster must already have its own colours).

Usage

```r
annotation_raster_tern(raster, xmin = 0, xmax = 1, ymin = 0,
    ymax = 1, interpolate = FALSE)
```

Arguments

- `raster`: raster object to display
- `xmin, xmax`: x location (in npc coordinates) giving horizontal location of raster
- `ymin, ymax`: y location (in npc coordinates) giving vertical location of raster
- `interpolate`: If TRUE interpolate linearly, if FALSE (the default) don’t interpolate.

Details

Most useful for adding bitmap images.

Author(s)

Nicholas Hamilton
Examples

data(Feldspar)
data(FeldsparRaster)
ggtern(Feldspar,aes(Ab,An,Or)) +
  theme_rgbw() +
  annotation_raster_tern(FeldsparRaster,xmin=0,xmax=1,ymin=0,ymax=1) +
  geom_mask() +
  geom_point(size=5,aes(shape=Feldspar,fill=Feldspar),color="black") +
  scale_shape_manual(values=c(21,24)) +
  labs(title="Demonstration of Raster Annotation")

---

Description

ggtern is a specialist extension to ggplot2 for rendering ternary diagrams, as such, many stats and geoms which come packaged with ggplot2 are either not relevant or will not work, as such, ggtern regulates during the plot construction process, which geoms and stats are able to be applied when using the coord_tern coordinate system. Attempting to apply non-approved geometries or stats (ie geometries / stats not in the below list), will result in the respective layers being stripped from the final plot.

Approved Geometries

The following geoms have been approved so far, including a combination of existing geoms and newly created geoms for the ggtern package APPROVED geoms in ggtern are as follows:

- geom_point
- geom_path
- geom_line
- geom_label
- geom_text
- geom_jitter
- geom_Tline
- geom_Rline
- geom_Lline
- geom_polygon
- geom_segment
- geom_count
- geom_errorbarT
- geom_errorbarL
- geom_errorbarR
• geom_density_tern
• geom_confidence
• geom_curve
• geom_mask
• geom_smooth_tern
• geom_blank
• geom_jitter
• geom_Tisoprop
• geom_Lisoprop
• geom_Risoprop
• geom_interpolate_tern
• geom_crosshair_tern
• geom_Tmark
• geom_Lmark
• geom_Rmark
• geom_point_swap
• geom_rect
• geom_polygon_closed
• geom_hex_tern
• geom_tri_tern
• geom_mean_ellipse
• geom_text_viewport
• geom_label_viewport

Approved Stats

The following stats have been approved so far, including a combination of existing stats and newly created stats for the ggtern package APPROVED stats in ggtern are as follows:

• stat_identity
• stat_confidence
• stat_density_tern
• stat_smooth_tern
• stat_sum
• stat_unique
• stat_interpolate_tern
• stat_mean_ellipse
• stat_hex_tern
• stat_tri_tern
Approved Positions

The following positions have been approved so far, including a combination of existing positions and newly created positions for the ggtern package APPROVED positions in ggternare as follows:

- position_identity
- position_nudge_tern
- position_jitter_tern

The balance of the available stats, geometries or positions within ggplot2 are either invalid or remain work in progress with regards to the ggtern package.

Author(s)

Nicholas Hamilton

---

**arrangeGrob**

Arrange multiple grobs on a page (ggtern version)

**Description**

A very slight modification to the original function, removing the explicit direction to use the ggplotGrob function from the ggplot2 namespace.

**Usage**

```r
arrangeGrob(..., grobs = list(...), layout_matrix, vp = NULL, name = "arrange", as.table = TRUE, respect = FALSE, clip = "off", nrow = NULL, ncol = NULL, widths = NULL, heights = NULL, top = NULL, bottom = NULL, left = NULL, right = NULL, padding = unit(0.5, "line"))
grid.arrange(..., newpage = TRUE)
```

**Arguments**

- `...`: grobs, gtables, ggplot or trellis objects
- `grobs`: list of grobs
- `layout_matrix`: optional layout
- `vp`: viewport
- `name`: argument of gtable
- `as.table`: logical: bottom-left to top-right (TRUE) or top-left to bottom-right (FALSE)
- `respect`: argument of gtable
- `clip`: argument of gtable
- `nrow`: argument of gtable
breaks_tern

### Generate Axis Breaks

**Description**

Calculates the Breaks for Major or Minor Gridlines based on the input limits.

**Usage**

```r
breaks_tern(limits = c(0, 1), isMajor = TRUE, n = 5)
```

**Arguments**

- `limits` the scale limits
- `isMajor` major or minor grids
- `n` number of breaks

**Examples**

```r
breaks_tern()
bREAKS_tern(limits = c(0,.5),FALSE,10)
```
Description
coord_tern is a function which creates a transformation mechanism between the ternary system, and, the cartesian system. It inherits from the fixed coordinate system, employing fixed ratio between x and y axes once transformed.

Usage
coord_tern(Tlim = NULL, Llim = NULL, Rlim = NULL, expand = TRUE)

Arguments
Tlim the range of T in the ternary space
Llim the range of L in the ternary space
Rlim the range of R in the ternary space
expand If TRUE, the default, adds a small expansion factor to the limits to ensure that data and axes don’t overlap. If FALSE, limits are taken exactly from the data or xlim/ylim.

Value
coord_tern returns a CoordTern ggproto

Aesthetics (Required in Each Layer)
coord_tern understands the following aesthetics (required aesthetics are in bold):

• x
• y
• z

Abovementioned limitations include the types of geometries which can be used (ie approved geometries), or modifications to required aesthetic mappings. One such essential patch is, for approved geometries previously requiring x and y coordinates, now require an additional z coordinate, and, geom_segment goes one step further in that it requires both an additional z and zend coordinate mappings.

In essence, the required aesthetics are the product between what is required of each 'layer' and what is required of the 'coordinate system'.

Author(s)
Nicholas Hamilton
Description

Data relating to Elkins and Groves Feldspar Data, the following datasets include the experimental data and sample raster data from one of the images in the referenced paper. **Feldspar** - Experimental Data **FeldsparRaster** - Raster Data for Fig. 6.

Usage

```r
#Experimental Data
data(Feldspar)

#Raster data
data(FeldsparRaster)
```

Format

**Feldspar** - One (1) row per Feldspar composition, **FeldsparRaster** - Raster Matrix

Author(s)

Nicholas Hamilton

References


See Also

Data

Examples

```r
#Summarize the Feldspar Data
data(Feldspar)
summary(Feldspar)

#Plot Felspar Data
ggtern(data=Feldspar,aes(x=An,y=Ab,z=Or)) + geom_point()

# Plot Feldspar data and Underlying Raster Image
data(FeldsparRaster)
ggtern(Feldspar,aes(Ab,An,Or)) + theme_rgbw() + annotation_raster_tern(FeldsparRaster,xmin=0,xmax=1,ymin=0,ymax=1) +
```
**data_Fragments**

```r
geom_point(size=5, aes(shape=Feldspar, fill=Feldspar), color='black') +
scale_shape_manual(values=c(21, 24)) +
labs(title = "Demonstration of Raster Annotation")
```

**data_Fragments**  
*Grantham and Valbel Rock Fragment Data*

**Description**

**ABSTRACT:** Chemical weathering influences the detrital composition of sand-size sediment derived from source areas subject to different amounts of precipitation in the Coweeta Basin, North Carolina. Of the grain types studied, rock fragments are most sensitive to chemical degradation; therefore, their abundance is the best indicator of cumulative weathering effects. Destruction of sand-size rock fragments by chemical weathering is a function of both the intensity and duration of chemical weathering experienced by grains in regoliths of the source area. In the Coweeta Basin, the intensity of chemical weathering is directly related to the climate via effective precipitation in individual subbasins, whereas the duration of chemical weathering is inversely related to the relief ratio of the watersheds. Therefore, soils in watersheds with low-relief ratios and high discharge per unit area experience the most extensive chemical weathering, and sediments derived from these watersheds contain the lowest percentage of rock fragments. The effects of climate alone cannot explain the systematic variation of rock fragment abundance in sediments from the Coweeta Basin. The compositional imprint left on these sediments by chemical weathering is a function of both climate and topographic slope in the sediment source area.

**Usage**

```r
data(Fragments)
```

**Format**

1 row per point. Each point contains data on the following:

1. **Watershed:** By id: 2, 10, 34, 41, 13, 27, 32 or 37,
2. **Position:** By name: Tallulah or Coweeta,
3. **CCWI:** The Cumulative Chemical Weathering Index: numeric
4. **Precipitation:** Average Annual Precipitation, numeric
5. **Discharge:** Annual Average Discharge, numeric
6. **Relief:** Relief Ratio, numeric
7. **GrainSize:** Coarse Medium or Fine,
8. **Sample:** Field Sampling, A, B or C
9. **Points:** The number of points measured for each sample
10. **Qm:** Multicrystalline Quarts Amount, percentage
11. **Qp:** Polycrystalline Quarts Amount, percentage
12. **Rf:** Rock Fragments Amount, percentage
13. **M:** Mica Amount, percentage
Author(s)
Jeremy Hummon Grantham and Michael Anthony Velbel

References

Examples
data(Fragments)
ggtern(Fragments,aes(Qm+Qp,Rf,M,colour=Sample)) +
  geom_density_tern(h=2,aes(fill=..level..),
    expand=0.75,alpha=0.5,bins=5) +
  geom_point(aes(shape=Position,size=Relief)) +
  theme_bw(base_size=8) +
  theme_showarrows() +
  custom_percent('%') +
  labs(title = "Grantham and Valbel Rock Fragment Data",
    x = "Q_{m+p}", xarrow = "Quartz (Multi + Poly)",
    y = "R_f", yarrow = "Rock Fragments",
    z = "M", zarrow = "Mica") +
  theme_latex() +
  facet_wrap(~Sample,nrow=2)

---

data_SkyeLava

Aichisons Skye Lavas

Description
AFM compositions of 23 aphyric Skye lavas.

Format
1 row per point, 23 points in total, Each point contains data on the following:

1. **No**: ID, S1 to S23
2. **A**: Percent Na2O+K2O
3. **F**: Percent Fe2O3
4. **F**: Percent MgO

Author(s)
J. Aitchison

References
Examples

# Emulate & Enhance plot produced in Fig. 3, pg 7 of:
# Martin-Fernandez, J.; Chacon-Duran, J. & Mateu-Figueras, G.
# Updating on the kernel density estimation for compositional data
# Proceedings of 17th Conference IASC-ERSS, Compstat, Roma,(Italy), 2006, 713-720

data(SkyeLava)
breaks = c(.01,.05,.10,.25,.5,.75,.9,.95,.99)
ggtern(SkyeLava,aes(F,A,M)) +
theme_bw() +
theme_showarrows() +
theme_latex() +
theme(tern.panel.grid.minor = element_blank(),
    tern.panel.grid.major = element_line(linetype='dotted',color='darkgray'),
    tern.axis.text = element_text(size=8)) +
geom_density_tern() +
geom_point() +
limit_tern(breaks = breaks,
    labels = sprintf("%.2f",breaks)) +
labs(title = "Aphyric Skye Lavas",
    subtitle = "AFM Compositions of 23 samples",
    Tarrow = "A = Na_2O + K_2O",
    Larrow = "F = Fe_20_3",
    Rarrow = "M = MgO")

---

data_USDA

USDA Textural Classification Data

Description

This dataset was issued by the United States Department of Agriculture (USDA) in the form of a ternary diagram, this original ternary diagram has been converted to numerical data and included here.

Usage

data(USDA)

Format

1 row per point, many points per classification representing the extremes of the area.

Author(s)

United States Department of Agriculture (USDA)
Nicholas Hamilton
Source
Soil Mechanics Level 1, Module 3, USDA Textural Classification Study Guide

See Also
ggtern datasets

Examples

```
#Load the Libraries
library(ggtern)
library(plyr)

#Load the Data.
data(USDA)

#Put tile labels at the midpoint of each tile.
USDA.LAB <- ddply(USDA,"Label",function(df){
  apply(df[,1:3],2,mean)
})

#Tweak
USDA.LAB$Angle = sapply(as.character(USDA.LAB$Label),function(x){
  switch(x,"Loamy Sand"=-35,0)
})

#Construct the plot.
ggtern(data=USDA,aes(Sand,Clay,Silt,color=Label,fill=Label)) +
  geom_polygon(alpha=0.75,size=0.5,color="black") +
  geom_mask() +
  geom_text(data=USDA.LAB,aes(label=Label,angle=Angle),color="black",size=3.5) +
  theme_rgbw() +
  theme_showsecondary() +
  theme_showarrows() +
  weight_percent() +
  guides(fill="none") +
  theme_legend_position("topleft") +
  labs(title = "USDA Textural Classification Chart",
       fill = "Textural Class",
       color = "Textural Class")
```

---

data_WhiteCells  
Aichisons White Cells

Description
White-cell compositions of 30 blood cells by two different methods
draw_key_tern

Format

1 row per point, 60 points in total, 2 experiments x 30 points each. Each point contains data on the following:

1. **No**: ID, S1 to S30
2. **Experiment**: MicroscopicInspection or ImageAnalysis
3. **G**: Fraction Granulocytes
4. **L**: Fraction Lymphocytes
5. **M**: Fraction Monocytes

Author(s)

J. Aitchison

References


Examples

data(WhiteCells)
ggtern(WhiteCells,aes(G,L,M)) +
geom_density_tern(aes(color=Experiment)) +
geom_point(aes(shape=Experiment)) +
facet_wrap(~Experiment,nrow=2)

---

draw_key_tern  Key drawing functions

Description

Each Geom has an associated function that draws the key when the geom needs to be displayed in a legend. These are the options built into ggplot2.

Usage

draw_key_crosshair_tern(data, params, size)
draw_key_Tmark(data, params, size)
draw_key_Lmark(data, params, size)
draw_key_Rmark(data, params, size)
draw_key_Tline(data, params, size)
draw_key_Lline(data, params, size)
draw_key_Rline(data, params, size)
draw_key_Tiso(data, params, size)
draw_key_Liso(data, params, size)
draw_key_Riso(data, params, size)
draw_key_point_swap(data, params, size)

Arguments

- **data**: A single row data frame containing the scaled aesthetics to display in this key.
- **params**: A list of additional parameters supplied to the geom.
- **size**: Width and height of key in mm.

Value

A grid grob.

Author(s)

Nicholas Hamilton

---

**geom_confidence_tern  Confidence Interval**

Description

Calculates the confidence intervals, via the Mahalanobis Distance and use of the Log-Ratio Transformation Statistic.

Usage

```r
geom_confidence_tern(mapping = NULL, data = NULL,
    stat = "ConfidenceTern", position = "identity", ....,
    lineend = "butt", linejoin = "round", linemitre = 1,
    na.rm = FALSE, show.legend = NA, inherit.aes = TRUE)
```

```r
stat_confidence_tern(mapping = NULL, data = NULL,
    geom = "ConfidenceTern", position = "identity", ....,
    contour = TRUE, n = 100, h = NULL, na.rm = FALSE,
    breaks = c(0.5, 0.9, 0.95), show.legend = NA, inherit.aes = TRUE)
```
Arguments

mapping Set of aesthetic mappings created by \texttt{aes()} or \texttt{aes()}. If specified and \texttt{inherit.aes} = \texttt{TRUE} (the default), it is combined with the default mapping at the top level of the plot. You must supply \texttt{mapping} if there is no plot mapping.

data The data to be displayed in this layer. There are three options:
If \texttt{NULL}, the default, the data is inherited from the plot data as specified in the call to \texttt{ggplot()}. A \texttt{data.frame}, or other object, will override the plot data. All objects will be fortified to produce a data frame. See \texttt{fortify()} for which variables will be created.
A function will be called with a single argument, the plot data. The return value must be a \texttt{data.frame}, and will be used as the layer data. A function can be created from a \texttt{formula} (e.g. \texttt{~ head(.x,10)}).

stat Use to override the default connection between \texttt{geom_smooth()} and \texttt{stat_smooth()}.

position Position adjustment, either as a string, or the result of a call to a position adjustment function.

\ldots Other arguments passed on to \texttt{layer()}. These are often aesthetics, used to set an aesthetic to a fixed value, like \texttt{colour = "red"} or \texttt{size = 3}. They may also be parameters to the paired geom/stat.

lineend Line end style (round, butt, square).

tlinejoin Line join style (round, mitre, bevel).

linemitre Line mitre limit (number greater than 1).
	na.rm If \texttt{FALSE}, the default, missing values are removed with a warning. If \texttt{TRUE}, missing values are silently removed.

show.legend logical. Should this layer be included in the legends? \texttt{NA}, the default, includes if any aesthetics are mapped. \texttt{FALSE} never includes, and \texttt{TRUE} always includes. It can also be a named logical vector to finely select the aesthetics to display.

inherit.aes If \texttt{FALSE}, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn’t inherit behaviour from the default plot specification, e.g. \texttt{borders()}.

geom Use to override the default connection between \texttt{geom_smooth()} and \texttt{stat_smooth()}.

contour If \texttt{TRUE}, contour the results of the 2d density estimation

n number of grid points in each direction

h Bandwidth (vector of length two). If \texttt{NULL}, estimated using \texttt{bandwidth.nrd}.

breaks the confidence intervals, default to 50, 90 and 95 percent.

Aesthetics

\texttt{geom_ConfidenceTern} understands the following aesthetics (required aesthetics are in bold):

\begin{itemize}
  \item \texttt{x}
  \item \texttt{y}
  \item \texttt{alpha}
\end{itemize}
• colour
• linetype
• size

Computed variables
Same as stat_contour

Author(s)
Nicholas Hamilton

Examples
data(Feldspar)
 ggtern(data=Feldspar,aes(An,Ab,Or)) +
  geom_point() +
  geom_confidence_tern()

Description
A new geometry, geom_crosshair_tern is one that that marks on the respective axes, the values of each data point. We also include additional geometries geom_Tmark, geom_Rmark and geom_Lmark – to render only the respective axis component of the abovementioned crosshair.

Usage
geom_crosshair_tern(mapping = NULL, data = NULL, stat = "identity",
position = "identity", ..., arrow = NULL, lineend = "butt",
na.rm = FALSE, show.legend = NA, inherit.aes = TRUE)

geom_Tmark(mapping = NULL, data = NULL, stat = "identity",
position = "identity", arrow = NULL, lineend = "butt",
na.rm = FALSE, show.legend = NA, inherit.aes = TRUE, ...)

geom_Lmark(mapping = NULL, data = NULL, stat = "identity",
position = "identity", arrow = NULL, lineend = "butt",
na.rm = FALSE, show.legend = NA, inherit.aes = TRUE, ...)

geom_Rmark(mapping = NULL, data = NULL, stat = "identity",
position = "identity", arrow = NULL, lineend = "butt",
na.rm = FALSE, show.legend = NA, inherit.aes = TRUE, ...)
Arguments

mapping
Set of aesthetic mappings created by `aes()` or `aes_()`. If specified and `inherit.aes = TRUE` (the default), it is combined with the default mapping at the top level of the plot. You must supply `mapping` if there is no plot mapping.

data
The data to be displayed in this layer. There are three options:
If `NULL`, the default, the data is inherited from the plot data as specified in the call to `ggplot()`.
A `data.frame`, or other object, will override the plot data. All objects will be fortified to produce a data frame. See `fortify()` for which variables will be created.
A function will be called with a single argument, the plot data. The return value must be a `data.frame`, and will be used as the layer data. A function can be created from a formula (e.g. `~ head(.x, 10)`).

stat
The statistical transformation to use on the data for this layer, as a string.

position
Position adjustment, either as a string, or the result of a call to a position adjustment function.

... Other arguments passed on to `layer()`. These are often aesthetics, used to set an aesthetic to a fixed value, like `colour = "red"` or `size = 3`. They may also be parameters to the paired geom/stat.

arrow specification for arrow heads, as created by `arrow()`.

lineend Line end style (round, butt, square).

na.rm If `FALSE`, the default, missing values are removed with a warning. If `TRUE`, missing values are silently removed.

show.legend logical. Should this layer be included in the legends? `NA`, the default, includes if any aesthetics are mapped. `FALSE` never includes, and `TRUE` always includes. It can also be a named logical vector to finely select the aesthetics to display.

inherit.aes If `FALSE`, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn’t inherit behaviour from the default plot specification, e.g. `borders()`.

Aesthetics

`geom_crosshair_tern` understands the following aesthetics (required aesthetics are in bold):

- x
- y
- z
- alpha
- colour
- linetype
- size

Author(s)

Nicholas Hamilton
Examples

```r
set.seed(1)
df = data.frame(x=runif(10), y=runif(10), z=runif(10))
base = ggtern(df, aes(x,y,z)) + geom_point()
base + geom_crosshair_tern()
base + geom_Tmark()
base + geom_Rmark()
base + geom_Lmark()
```

---

**geom_density_tern**

**Density Estimate (ggtern version)**

**Description**

Perform a 2D kernel density estimation using kde2d and display the results with contours. This can be useful for dealing with overplotting. Additional weight aesthetic (see aesthetic section below) permits better weighting if desired.

**Usage**

```r
geom_density_tern(mapping = NULL, data = NULL, stat = "DensityTern",
                  position = "identity", ..., lineend = "butt", linejoin = "round",
                  linemitre = 1, na.rm = FALSE, show.legend = NA,
                  inherit.aes = TRUE)
```

```r
stat_density_tern(mapping = NULL, data = NULL, geom = "density_tern",
                  position = "identity", ..., contour = TRUE, n = 100, h = NULL,
                  bdl = 0, bdl.val = NA, na.rm = FALSE, show.legend = NA,
                  inherit.aes = TRUE, weight = 1, base = "ilr", expand = c(0.5, 0.5))
```

**Arguments**

- **mapping**
  Set of aesthetic mappings created by `aes()` or `aes()`. If specified and `inherit.aes = TRUE` (the default), it is combined with the default mapping at the top level of the plot. You must supply `mapping` if there is no plot mapping.

- **data**
  The data to be displayed in this layer. There are three options:
  - If `NULL`, the default, the data is inherited from the plot data as specified in the call to `ggplot()`.
  - A `data.frame`, or other object, will override the plot data. All objects will be fortified to produce a data frame. See `fortify()` for which variables will be created.
  - A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. `~ head(.x,10)`).

- **stat**
  The statistical transformation to use on the data for this layer, as a string.
**geom_density_tern**

- **position**: Position adjustment, either as a string, or the result of a call to a position adjustment function.
- **...**: Other arguments passed on to `layer()`. These are often aesthetics, used to set an aesthetic to a fixed value, like `colour = "red"` or `size = 3`. They may also be parameters to the paired geom/stat.
- **lineend**: Line end style (round, butt, square).
- **linejoin**: Line join style (round, mitre, bevel).
- **linemitre**: Line mitre limit (number greater than 1).
- **na.rm**: If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.
- **show.legend**: logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.
- **inherit.aes**: If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn’t inherit behaviour from the default plot specification, e.g. `borders()`.
- **geom**: Use to override the default connection between `geom_density_2d` and `stat_density_2d`.
- **contour**: If TRUE, contour the results of the 2d density estimation.
- **n**: Number of grid points in each direction.
- **h**: Bandwidth (vector of length two) as a multiple of the best estimate, estimated using `bandwidth.nrd`.
- **bdl**: the threshold for detection limit. This is applied against the output of `acomp` function, so it is expected as a fraction in the range [0,1]
- **bdl.val**: compositions which have components that are below the detection limit, will have these components replaced by this val. If it is NA then these items will be discarded. If the value is something other than 'NA', then all values less than `bdl` will be replaced and therefore included in the final density estimate.
- **weight**: weighting for weighted kde2d estimate, default’s to 1, which is non-weighted and equivalent to the usual kde2d calculation
- **base**: the base transformation of the data, options include ’identity’ (ie direct on the cartesian space), or ’ilr’ which means to use the isometric log ratio transformation.
- **expand**: Calculate on a mesh which extends beyond the grid of the plot region by this amount If NULL, estimated using `bandwidth.nrd`.

**Aesthetics**

`geom_density_tern` understands the following aesthetics (required aesthetics are in bold):

- **x**
- **y**
- **alpha**
- **colour**
• linetype
• size
• weight

Author(s)
Nicholas Hamilton
Nicholas Hamilton

Examples

```r
# Plot Density Estimate, on isometric log ratio transformation of original data
data('Feldspar')
ggtern(Feldspar,aes(Ab,An,Or)) +
  geom_density_tern(aes(color=..level..),bins=5) +
  geom_point()

# Plot Density Estimate w/ Polygon Geometry
data('Feldspar')
ggtern(data=Feldspar,aes(Ab,An,Or)) +
  stat_density_tern(geom='polygon',
    aes(fill=..level..),
    bins=5,
    color='grey') +
  geom_point()
```

---

**geom_errorbarX**

Ternary Error Bars

Description

`geom_errorbarT`, `geom_errorbarL` and `geom_errorbarR` are geometries to render error bars for the top, left and right apex species respectively, analogous to `geom_errorbar` and/or `geom_errorbarh` as provided in the base ggplot2 package.

Usage

```r
geom_errorbarT(mapping = NULL, data = NULL, stat = "identity",
  position = "identity", ..., arrow = NULL, lineend = "butt",
  na.rm = FALSE, show.legend = NA, inherit.aes = TRUE)

geom_errorbarL(mapping = NULL, data = NULL, stat = "identity",
  position = "identity", arrow = NULL, lineend = "butt",
  na.rm = FALSE, show.legend = NA, inherit.aes = TRUE, ...)

geom_errorbarR(mapping = NULL, data = NULL, stat = "identity",
  position = "identity", arrow = NULL, lineend = "butt",
  na.rm = FALSE, show.legend = NA, inherit.aes = TRUE, ...)
```
Arguments

mapping

Set of aesthetic mappings created by \texttt{aes()} or \texttt{aes()} . If specified and \texttt{inherit.aes = TRUE} (the default), it is combined with the default mapping at the top level of the plot. You must supply \texttt{mapping} if there is no plot mapping.

data

The data to be displayed in this layer. There are three options:

If \texttt{NULL}, the default, the data is inherited from the plot data as specified in the call to \texttt{ggplot()}. A \texttt{data.frame}, or other object, will override the plot data. All objects will be fortified to produce a data frame. See \texttt{fortify()} for which variables will be created.

A function will be called with a single argument, the plot data. The return value must be a \texttt{data.frame}, and will be used as the layer data. A function can be created from a formula (e.g. \texttt{~ head(.x,10)}).

stat

The statistical transformation to use on the data for this layer, as a string.

position

Position adjustment, either as a string, or the result of a call to a position adjustment function.

... Other arguments passed on to \texttt{layer()}. These are often aesthetics, used to set an aesthetic to a fixed value, like \texttt{colour = "red"} or \texttt{size = 3}. They may also be parameters to the paired geom/stat.

arrow specification for arrow heads, as created by \texttt{arrow}().

lineend Line end style (round, butt, square).

na.rm If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.

show.legend If FALSE, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.

inherit.aes If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn’t inherit behaviour from the default plot specification, e.g. \texttt{borders()}.

Aesthetics (\texttt{geom_errorbart})

\texttt{geom_errorbart} understands the following aesthetics (required aesthetics are in bold):

• \texttt{Tmax}
• \texttt{Tmin}
• \texttt{x}
• \texttt{y}
• \texttt{z}
• \texttt{alpha}
• \texttt{colour}
• \texttt{linetype}
• \texttt{size}
Aesthetics (geom_errorbarL)

geom_errorbarL understands the following aesthetics (required aesthetics are in bold):

- Lmax
- Lmin
- x
- y
- z
- alpha
- colour
- linetype
- size

Aesthetics (geom_errorbarR)

geom_errorbarR understands the following aesthetics (required aesthetics are in bold):

- Rmax
- Rmin
- x
- y
- z
- alpha
- colour
- linetype
- size

Author(s)

Nicholas Hamilton

Examples

```r
# Example with Dummy Data.
tmp <- data.frame(x=1/3,
y=1/3,
z=1/3,
Min=1/3-1/6,
Max=1/3+1/6)
ggtern(data=tmp,aes(x,y,z)) +
  geom_point() +
  geom_errorbarT(aes(Tmin=Min,Tmax=Max),colour='red') +
  geom_errorbarL(aes(Lmin=Min,Lmax=Max),colour='green') +
  geom_errorbarR(aes(Rmin=Min,Rmax=Max),colour='blue')
```
Hexbin (ggtern version).

Description

Divides the plane into regular hexagons, counts the number of cases in each hexagon, and then (by default) maps the number of cases to the hexagon fill. Hexagon bins avoid the visual artefacts sometimes generated by the very regular alignment of [geom_bin2d()].

Usage

```r
geom_hex_tern(mapping = NULL, data = NULL, stat = "hex_tern",
              position = "identity", ..., fun = sum, na.rm = FALSE,
              show.legend = NA, inherit.aes = TRUE)
stat_hex_tern(mapping = NULL, data = NULL, geom = "hex_tern",
              position = "identity", ..., bins = 30, fun = sum,
              binwidth = NULL, na.rm = FALSE, show.legend = NA,
              inherit.aes = TRUE)
```

Arguments

- `mapping` Set of aesthetic mappings created by `aes()` or `aes()`. If specified and `inherit.aes` = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply `mapping` if there is no plot mapping.
- `data` The data to be displayed in this layer. There are three options:
  - If NULL, the default, the data is inherited from the plot data as specified in the call to `ggplot()`.
  - A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See `fortify()` for which variables will be created.
  - A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. `~ head(.x,10)`).
- `position` Position adjustment, either as a string, or the result of a call to a position adjustment function.
- `...` Other arguments passed on to `layer()`. These are often aesthetics, used to set an aesthetic to a fixed value, like `colour = "red"` or `size = 3`. They may also be parameters to the paired geom/stat.
- `fun` the scalar function to use for the statistic.
- `na.rm` If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.
- `show.legend` logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.
**geom_hex_tern**

- **inherit.aes**
  - If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn’t inherit behaviour from the default plot specification, e.g. `borders()`.

- **geom, stat**
  - Override the default connection between 'geom_hex_tern' and 'stat_hex_tern'

- **bins**
  - numeric vector giving number of bins in both vertical and horizontal directions. Set to 30 by default.

- **binwidth**
  - Numeric vector giving bin width in both vertical and horizontal directions. Overrides bins if both set.

**Details**

This geometry is loosely based on the base ggplot2 geom_hex, with a few subtle (but advantageous differences). The user can control the border thickness of the hexagonal polygons using the size aesthetic. The user can also control the particular statistic to use, by defining the `fun` argument (sum by default), which by default is applied over a value of 1 per point, however, this can also be mapped to a data variable via the `value` mapping.

**Aesthetics**

@section Aesthetics: `geom_hex()` understands the following aesthetics (required aesthetics are in bold):

- x
- y
- alpha
- colour
- fill
- group
- linetype
- size

Learn more about setting these aesthetics in vignette("ggplot2-specs").

**Examples**

```r
set.seed(1)
n = 1000
df = data.frame(x = runif(n),
                 y = runif(n),
                 z = runif(n),
                 wt = runif(n))

#Equivalent of Hexbin
ggtern(df,aes(x,y,z)) +
       geom_hex_tern(binwidth=0.1)

#Calculate Mean of variable wt
ggtern(df,aes(x,y,z)) +
```
# geom_interpolate_tern

```r
geom_hex_tern(binwidth=0.05,
        aes(value=wt),
        fun=mean)
```

```r
# Custom functions, for ex. discrete output...
myfun = function(x) sample(LETTERS,1)
``` ggtern(df,aes(x,y,z)) +

```r
geom_hex_tern(binwidth=0.05,
        fun=myfun)
```

## geom_interpolate_tern  Ternary Interpolation

### Description

This is the heavily requested geometry for interpolating between ternary values, results being rendered using contours on a ternary mesh.

### Usage

```r
geom_interpolate_tern(mapping = NULL, data = NULL, 
stat = "InterpolateTern", position = "identity", ..., 
method = "auto", formula = value ~ poly(x, y, degree = 1), 
lineend = "butt", linejoin = "round", linemitre = 1, 
na.rm = FALSE, show.legend = NA, inherit.aes = TRUE)
```

```r
stat_interpolate_tern(mapping = NULL, data = NULL, 
geom = "interpolate_tern", position = "identity", ..., 
method = "auto", na.rm = FALSE, show.legend = NA, 
inherit.aes = TRUE, n = 80, formula = value ~ poly(x, y, degree = 1), base = "ilr")
```

### Arguments

- **mapping**
  - Set of aesthetic mappings created by `aes()` or `aes()`. If specified and `inherit.aes` = `TRUE` (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.

- **data**
  - The data to be displayed in this layer. There are three options:
    - If `NULL`, the default, the data is inherited from the plot data as specified in the call to `ggplot()`.
    - A `data.frame`, or other object, will override the plot data. All objects will be fortified to produce a data frame. See `fortify()` for which variables will be created.
    - A function will be called with a single argument, the plot data. The return value must be a `data.frame`, and will be used as the layer data. A function can be created from a `formula` (e.g. `~ head(.x, 10)`).
stat
Use to override the default connection between geom_smooth() and stat_smooth().

position
Position adjustment, either as a string, or the result of a call to a position adjustment function.

... Other arguments passed on to layer(). These are often aesthetics, used to set an aesthetic to a fixed value, like colour = "red" or size = 3. They may also be parameters to the paired geom/stat.

method
Smoothing method (function) to use, accepts either NULL or a character vector, e.g. "lm", "glm", "gam", "loess" or a function, e.g. MASS::rlm or mgcv::gam, stats::lm, or stats::loess. "auto" is also accepted for backwards compatibility. It is equivalent to NULL.

For method = NULL the smoothing method is chosen based on the size of the largest group (across all panels). stats::loess() is used for less than 1,000 observations; otherwise mgcv::gam() is used with formula = y ~ s(x, bs = "cs") with method = "REML". Somewhat anecdotally, loess gives a better appearance, but is $O(N^2)$ in memory, so does not work for larger datasets.

If you have fewer than 1,000 observations but want to use the same gam() model that method = NULL would use, then set method = "gam", formula = y ~ s(x, bs = "cs").

formula
Formula to use in smoothing function, eg. y ~ x, y ~ poly(x,2), y ~ log(x).
NULL by default, in which case method = NULL implies formula = y ~ x when there are fewer than 1,000 observations and formula = y ~ s(x, bs = "cs") otherwise.

lineend
Line end style (round, butt, square).

linejoin
Line join style (round, mitre, bevel).

linemitre
Line mitre limit (number greater than 1).

na.rm
If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.

show.legend
logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.

inherit.aes
If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn’t inherit behaviour from the default plot specification, e.g. borders().

geom
Use to override the default connection between geom_smooth() and stat_smooth().

n
number of grid points in each direction

base
the base transformation of the data, options include 'identity' (ie direct on the cartesian space), or 'ilr' which means to use the isometric log ratio transformation.

Aesthetics
geom_InterpolateTern understands the following aesthetics (required aesthetics are in bold):

• x
• y
• alpha
• colour
• linetype
• size

Author(s)
Nicholas Hamilton

Examples

data(Feldspar)
ggtern(Feldspar,aes(Ab,An,Or,value=T.C)) +
stat_interpolate_tern(geom="polygon",
  formula=value~x+y,
  method=lm,n=100,
  breaks=seq(0,1000,by=100),
  aes(fill=..level..),expand=1) +
geom_point()
A `data.frame`, or other object, will override the plot data. All objects will be fortified to produce a data frame. See `fortify()` for which variables will be created.

A function will be called with a single argument, the plot data. The return value must be a `data.frame`, and will be used as the layer data. A function can be created from a formula (e.g. `~ head(.x, 10)`).

**stat**  The statistical transformation to use on the data for this layer, as a string.

**position**  Position adjustment, either as a string, or the result of a call to a position adjustment function.

**...**  Other arguments passed on to `layer()`. These are often aesthetics, used to set an aesthetic to a fixed value, like `colour = "red"` or `size = 3`. They may also be parameters to the paired geom/stat.

hjust  horizontal justification

vjust  vertical justification

**parse**  If TRUE, the labels will be parsed into expressions and displayed as described in ?plotmath.

**label.padding**  Amount of padding around label. Defaults to 0.25 lines.

**label.r**  Radius of rounded corners. Defaults to 0.15 lines.

**label.size**  Size of label border, in mm.

**na.rm**  If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.

**show.legend**  logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.

**inherit.aes**  If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn’t inherit behaviour from the default plot specification, e.g. `borders()`.

### Aesthetics

`geom_label` understands the following aesthetics (required aesthetics are in bold):

- **label**
- **x**
- **y**
- **alpha**
- **angle**
- **colour**
- **family**
- **fill**
- **fontface**
- **hjust**
- **lineheight**
- **size**
- **vjust**
**geom_mask**

**Author(s)**
Nicholas Hamilton

**See Also**
geom_label

**Examples**

```r
library(ggplot2)
data(Feldspar)
base = ggtern(data=Feldspar,aes(Ab,An,Or)) +
  geom_mask() +
  geom_point() +
  geom_label_viewport(x=0.5,y=0.5,label="Middle",color='red') +
  geom_label_viewport(x=1.0,y=1.0,label="Top Right",color='blue') +
  geom_label_viewport(x=0.0,y=0.0,label="Bottom Left",color='green') +
  geom_label_viewport(x=0.0,y=1.0,label="Top Left",color='orange') +
  geom_label_viewport(x=1.0,y=0.0,label="Bottom Right",color='magenta')
base

base +
  geom_label_viewport(x=0.9,y=0.5,label="Clipping Turned Off",color='purple',hjust=0,clip='on')

base +
  geom_label_viewport(x=0.9,y=0.5,label="Clipping Turned Off",color='purple',hjust=0,clip='off')
```

---

**geom_mask**  
*Apply Manual Clipping Mask*

**Description**

This function creates a manual clipping mask, which in turn suppresses the standard clipping mask that would otherwise be rendered in the foreground rendering procedure, giving the user control over the exact placement with respect to other layers. For example, the user may wish to have the clipping mask placed after the `geom_point(...)` layer, but before the `geom_label(...)` layer, this situation has been demonstrated in the example below. In the event that the user wishes to suppress the mask altogether, then a convenience function has been provided, `theme_nomask()`.

**Usage**

```r
geom_mask()
```

**Author(s)**
Nicholas Hamilton
Examples

data(Feldspar)
x = ggtern(Feldspar,aes(Ab,An,Or,label=Experiment)) + geom_point()

#Default Behaviour
x + geom_label()

#Insert manual mask before the labels, to prevent them being truncated
x + geom_point(size=6) + geom_mask() + geom_label()

geom_mean_ellipse

Mean Ellipse

Description

Produce ellipses from a mean and a variance of ternary compositional data, based off the function included in the compositions package.

Usage

geom_mean_ellipse(mapping = NULL, data = NULL, stat = "MeanEllipse",
position = "identity", ..., lineend = "butt", linejoin = "round",
linemitre = 1, na.rm = FALSE, show.legend = NA,
inherit.aes = TRUE)

stat_mean_ellipse(mapping = NULL, data = NULL, geom = "MeanEllipse",
position = "identity", ..., steps = 72, r = 1, na.rm = FALSE,
show.legend = NA, inherit.aes = TRUE)

Arguments

mapping Set of aesthetic mappings created by aes() or aes(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.

data The data to be displayed in this layer. There are three options:
If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot().
A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created.
A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x,10)).

stat Use to override the default connection between geom_smooth() and stat_smooth().

position Position adjustment, either as a string, or the result of a call to a position adjustment function.
Other arguments passed on to `layer()`. These are often aesthetics, used to set an aesthetic to a fixed value, like colour = "red" or size = 3. They may also be parameters to the paired geom/stat.

- `lineend` Line end style (round, butt, square).
- `linejoin` Line join style (round, mitre, bevel).
- `linemitre` Line mitre limit (number greater than 1).
- `na.rm` If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.
- `show.legend` logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.
- `inherit.aes` If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn’t inherit behaviour from the default plot specification, e.g. `borders()`.
- `geom` Use to override the default connection between `geom_smooth()` and `stat_smooth()`.
- `steps` the number of discretisation points to draw the ellipses
- `r` a scaling of the half-diameters

### Aesthetics

`geom_MeanEllipse` understands the following aesthetics (required aesthetics are in bold):

- `x`
- `y`
- `alpha`
- `colour`
- `linetype`
- `size`

### Computed variables

Same as `stat_contour`

### Author(s)

Nicholas Hamilton & Ashton Drew

### Examples

```r
data(Feldspar)
ggtern(data=Feldspar,aes(An,Ab,Or)) + geom_point() + geom_mean_ellipse()
data(Feldspar)
ggtern(data=Feldspar,aes(An,Ab,Or)) + theme_bw() + geom_point()
```
Description

The geom_point_swap geometry is used to create scatterplots, however, this version swaps the colour and the fill mappings. Useful if the fill mapping is already occupied (say with existing polygon geometry), this geometry will allow points of shape 21-25 to use colour mapping for the center colour, and fill mapping for the border.

Usage

geom_point_swap(mapping = NULL, data = NULL, stat = "identity", position = "identity", ..., na.rm = FALSE, show.legend = NA, inherit.aes = TRUE)

Arguments

mapping
Set of aesthetic mappings created by aes() or aes_. If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.

data
The data to be displayed in this layer. There are three options:
If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot().
A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created.
A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x,10)).

stat
The statistical transformation to use on the data for this layer, as a string.

position
Position adjustment, either as a string, or the result of a call to a position adjustment function.

...
Other arguments passed on to layer(). These are often aesthetics, used to set an aesthetic to a fixed value, like colour = "red" or size = 3. They may also be parameters to the paired geom/stat.

na.rm
If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.

show.legend
logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.

inherit.aes
If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn’t inherit behaviour from the default plot specification, e.g. borders().
Author(s)

Nicholas Hamilton

Examples

data(Feldspar)
ggtern(Feldspar,aes(Ab,An,Or)) +
stat_confidence_tern(geom='polygon',aes(fill=.level..),color='white') +
geom_mask() +
geom_point_swap(aes(colour=T.C,shape=Feldspar),fill='black',size=5) +
scale_shape_manual(values=c(21,24)) +
scale_color_gradient(low='green',high='red') +
labs(title='Feldspar',color='Temperature',fill='Confidence')

Description

A little like geom_area, in the sense that polygons are either upper or lower closed based on the
starting and finishing points index.

Usage

geom_polygon_closed(mapping = NULL, data = NULL, stat = "identity",
position = "identity", ..., na.rm = FALSE, show.legend = NA,
inherit.aes = TRUE, closure = "none")

Arguments

mapping Set of aesthetic mappings created by aes() or aes_.() If specified and inherit.aes
= TRUE (the default), it is combined with the default mapping at the top level of
the plot. You must supply mapping if there is no plot mapping.
data The data to be displayed in this layer. There are three options:
If NULL, the default, the data is inherited from the plot data as specified in the
call to ggplot().
A data.frame, or other object, will override the plot data. All objects will be
fortified to produce a data frame. See fortify() for which variables will be
created.
A function will be called with a single argument, the plot data. The return
value must be a data.frame, and will be used as the layer data. A function
can be created from a formula (e.g. ~ head(.x,10)).
stat The statistical transformation to use on the data for this layer, as a string.
position Position adjustment, either as a string, or the result of a call to a position adjustment function.
... Other arguments passed on to layer(). These are often aesthetics, used to set an aesthetic to a fixed value, like colour = "red" or size = 3. They may also be parameters to the paired geom/stat.

na.rm If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.

show.legend logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.

inherit.aes If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn’t inherit behaviour from the default plot specification, e.g. borders().

closure one of 'none', 'upper' or 'lower'

Author(s)
Nicholas Hamilton

geom_smooth_tern

Add a Smoothed Conditional Mean.

Description
Aids the eye in seeing patterns in the presence of overplotting. geom_smooth_tern and stat_smooth_tern are effectively aliases: they both use the same arguments. Use geom_smooth_tern unless you want to display the results with a non-standard geom.

Usage

geom_smooth_tern(mapping = NULL, data = NULL, position = "identity",
... , method = "auto", formula = y ~ x, se = TRUE, na.rm = FALSE,
show.legend = NA, inherit.aes = TRUE, expand = c(0.5, 0.5))

stat_smooth_tern(mapping = NULL, data = NULL, position = "identity",
... , method = "auto", formula = y ~ x, se = TRUE, n = 80,
span = 0.75, fullrange = FALSE, level = 0.95,
method.args = list(), na.rm = FALSE, show.legend = NA,
inherit.aes = TRUE, expand = c(0.5, 0.5))

Arguments

mapping Set of aesthetic mappings created by aes() or aes_(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data The data to be displayed in this layer. There are three options:

If NULL, the default, the data is inherited from the plot data as specified in the call to `ggplot()`.

A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See `fortify()` for which variables will be created.

A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. `~ head(.x,10)`).

position Position adjustment, either as a string, or the result of a call to a position adjustment function.

... Other arguments passed on to `layer()`. These are often aesthetics, used to set an aesthetic to a fixed value, like `colour = "red"` or `size = 3`. They may also be parameters to the paired geom/stat.

method Smoothing method (function) to use, accepts either NULL or a character vector, e.g. "lm", "glm", "gam", "loess" or a function, e.g. MASS::rlm or mgcv::gam, stats::lm, or stats::loess. "auto" is also accepted for backwards compatibility. It is equivalent to NULL.

For `method = NULL` the smoothing method is chosen based on the size of the largest group (across all panels). `stats::loess()` is used for less than 1,000 observations; otherwise `mgcv::gam()` is used with formula = `y ~ s(x,bs = "cs")` with `method = "REML"`. Somewhat anecdotally, loess gives a better appearance, but is $O(N^2)$ in memory, so does not work for larger datasets.

If you have fewer than 1,000 observations but want to use the same `gam()` model that `method = NULL` would use, then set `method = "gam", formula = y ~ s(x, bs = "cs")`.

formula Formula to use in smoothing function, eg. `y ~ x`, `y ~ poly(x,2)`, `y ~ log(x)`. NULL by default, in which case `method = NULL` implies formula = `y ~ x` when there are fewer than 1,000 observations and `formula = y ~ s(x,bs = "cs")` otherwise.

se Display confidence interval around smooth? (TRUE by default, see `level` to control.)

na.rm If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.

show.legend logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.

inherit.aes If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn’t inherit behaviour from the default plot specification, e.g. `borders()`.

expand expand the range of values by this much (vector of length 2) when `fullrange` is set to TRUE.

n Number of points at which to evaluate smoother.

span Controls the amount of smoothing for the default loess smoother. Smaller numbers produce wigglier lines, larger numbers produce smoother lines.
fullrange  Should the fit span the full range of the plot, or just the data?
level      Level of confidence interval to use (0.95 by default).
method.args List of additional arguments passed on to the modelling function defined by method.

Author(s)
Nicholas Hamilton

Examples

data(Feldspar)
ggtern(data=Feldspar,aes(Ab,An,Or,group=Feldspar)) +
  geom_smooth_tern(method=lm,fullrange=TRUE,colour='red') +
  geom_point() +
  labs(title="Example Smoothing")

geom_text_viewport  Draw Text at Relative Position on Viewport

Description
Since it is sometimes counter intuitive for working with ternary or other non-cartesian coordinates in
the event that the user wishes to place a text-geometry based on visual inspection, this geometry
positions such text item at a fraction from x=[0,1] and y=[0,1] of the viewport in x and y cartesian
coordinates.

Usage

ggeom_text_viewport(mapping = NULL, data = NULL, stat = "identity",
  position = "identity", ..., hjust = "inward", vjust = "inward",
  parse = FALSE, check_overlap = FALSE, na.rm = FALSE,
  show.legend = NA, inherit.aes = TRUE)

Arguments

  mapping      Set of aesthetic mappings created by aes() or aes_.() If specified and inherit.aes
               = TRUE (the default), it is combined with the default mapping at the top level of
               the plot. You must supply mapping if there is no plot mapping.
  data         The data to be displayed in this layer. There are three options:
               If NULL, the default, the data is inherited from the plot data as specified in the
               call to ggplot().
               A data.frame, or other object, will override the plot data. All objects will be
               fortified to produce a data frame. See fortify() for which variables will be
               created.
               A function will be called with a single argument, the plot data. The return
               value must be a data.frame, and will be used as the layer data. A function
               can be created from a formula (e.g. ~ head(.x,10)).
stat
The statistical transformation to use on the data for this layer, as a string.

position
Position adjustment, either as a string, or the result of a call to a position adjustment function.

... Other arguments passed on to layer(). These are often aesthetics, used to set an aesthetic to a fixed value, like colour = "red" or size = 3. They may also be parameters to the paired geom/stat.

hjust
horizontal justification

vjust
vertical justification

parse
If TRUE, the labels will be parsed into expressions and displayed as described in ?plotmath.

check_overlap
If TRUE, text that overlaps previous text in the same layer will not be plotted. check_overlap happens at draw time and in the order of the data. Therefore data should be arranged by the label column before calling geom_label() or geom_text().

na.rm
If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.

show.legend
logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.

inherit.aes
If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn’t inherit behaviour from the default plot specification, e.g. borders().

Aesthetics

geom_Text understands the following aesthetics (required aesthetics are in bold):

- label
- x
- y
- alpha
- angle
- colour
- family
- fontface
- hjust
- lineheight
- size
- vjust

Author(s)
Nicholas Hamilton
geom_tri_tern

**Tribin (ggtern version).**

**Description**

Divides the plane into regular triangles, counts the number of cases in each triangles, and then (by default) maps the number of cases to the triangle fill.

**Usage**

```r
geom_tri_tern(mapping = NULL, data = NULL, stat = "tri_tern",  
position = "identity", ..., fun = sum, na.rm = FALSE,  
show.legend = NA, inherit.aes = TRUE)
```

```r
stat_tri_tern(mapping = NULL, data = NULL, geom = "tri_tern",  
position = "identity", ..., bins = 30, fun = sum,  
centroid = FALSE, na.rm = FALSE, show.legend = NA,  
inherit.aes = TRUE)
```

**Arguments**

- `mapping` Set of aesthetic mappings created by `aes()` or `aes()`. If specified and `inherit.aes` = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply `mapping` if there is no plot mapping.
**geom_tri_tern**

The data to be displayed in this layer. There are three options:

If `NULL`, the default, the data is inherited from the plot data as specified in the call to `ggplot()`.

A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See `fortify()` for which variables will be created.

A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. `~ head(.x, 10)`).

**position**

Position adjustment, either as a string, or the result of a call to a position adjustment function.

... Other arguments passed on to `layer()`. These are often aesthetics, used to set an aesthetic to a fixed value, like `colour = "red"` or `size = 3`. They may also be parameters to the paired geom/stat.

**fun**

the scalar function to use for the statistic

**na.rm**

If `FALSE`, the default, missing values are removed with a warning. If `TRUE`, missing values are silently removed.

**show.legend**

logical. Should this layer be included in the legends? `NA`, the default, includes if any aesthetics are mapped. `FALSE` never includes, and `TRUE` always includes. It can also be a named logical vector to finely select the aesthetics to display.

**inherit.aes**

If `FALSE`, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn’t inherit behaviour from the default plot specification, e.g. `borders()`.

**geom, stat**

Override the default connection between 'geom_hex_tern' and 'stat_hex_tern'

**bins**

numeric vector giving number of bins in both vertical and horizontal directions. Set to 30 by default.

**centroid**

logical to return the centroid of the polygon, rather than the complete polygon

---

**Aesthetics**

@section Aesthetics: geom_hex() understands the following aesthetics (required aesthetics are in bold):

- x
- y
- alpha
- colour
- fill
- group
- linetype
- size

Learn more about setting these aesthetics in vignette("ggplot2-specs").
Examples

```r
set.seed(1)
n = 1000
df = data.frame(x = runif(n),
y = runif(n),
z = runif(n),
wt = runif(n))
# Equivalent of Hexbin
ggtern(df,aes(x,y,z)) +
  geom_tri_tern(bins=10,aes(fill=..count..)) +
  geom_point(size=0.25)

# Custom Function, Mean
#ggtern(df,aes(x,y,z)) +
ggtern(df,aes(x,y,z)) +
  geom_tri_tern(bins=5,aes(fill=..stat..,value=wt),fun=mean) +
  geom_point(size=0.25)
```

---

**geom_Xisoprop**  
*Fixed Value Isoproportion Lines*

Description

Create fixed isoproportion lines for each of the ternary axes, `geom_Xisoprop(...)`, `(X = T, L, R)` will draw an isoproportion line projecting from the T, L and R apex respectively.

Usage

```r
geom_Tisoprop(mapping = NULL, data = NULL, ..., value, na.rm = FALSE, show.legend = NA)
geom_Lisoprop(mapping = NULL, data = NULL, ..., value, na.rm = FALSE, show.legend = NA)
geom_Risoprop(mapping = NULL, data = NULL, ..., value, na.rm = FALSE, show.legend = NA)
```

Arguments

- **mapping**  
  Set of aesthetic mappings created by `aes()` or `aes()`. If specified and `inherit.aes` = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply `mapping` if there is no plot mapping.

- **data**  
  The data to be displayed in this layer. There are three options:
  - If NULL, the default, the data is inherited from the plot data as specified in the call to `ggplot()`.
  - A `data.frame`, or other object, will override the plot data. All objects will be fortified to produce a data frame. See `fortify()` for which variables will be created.
A function will be called with a single argument, the plot data. The return value must be a `data.frame`, and will be used as the layer data. A function can be created from a formula (e.g. `~ head(.x, 10)`).

Other arguments passed on to `layer()`. These are often aesthetics, used to set an aesthetic to a fixed value, like `colour = "red"` or `size = 3`. They may also be parameters to the paired geom/stat.

- `value`, the isoproportion ratio to draw
- `na.rm` If `FALSE`, the default, missing values are removed with a warning. If `TRUE`, missing values are silently removed.
- `show.legend` logical. Should this layer be included in the legends? `NA`, the default, includes if any aesthetics are mapped. `FALSE` never includes, and `TRUE` always includes. It can also be a named logical vector to finely select the aesthetics to display.

**Aesthetics**

`geom_Tisoprop` understands the following aesthetics (required aesthetics are in bold):

- `value`
- `alpha`
- `arrow`
- `colour`
- `linetype`
- `size`

**Author(s)**

Nicholas Hamilton

**Examples**

```r
data(Feldspar)
ggtern(data=Feldspar, aes(Ab, An, Or)) +
geom_Tisoprop(value=0.5) +
geom_Lisoprop(value=0.5) +
geom_Risoprop(value=0.5) +
geom_point()
```

---

**geom_Xline**

**Fixed Value Lines**

**Description**

Plot fixed value lines, for the top, left and right axis, analogous to the `geom_hline` and `geom_vline` geometries in `ggplot2`
Usage

```r
geom_Tline(mapping = NULL, data = NULL, ..., Tintercept, na.rm = FALSE, show.legend = NA)
Tline(mapping = NULL, data = NULL, ..., Tintercept, na.rm = FALSE, show.legend = NA)
tline(mapping = NULL, data = NULL, ..., Tintercept, na.rm = FALSE, show.legend = NA)
geom_Lline(mapping = NULL, data = NULL, ..., Lintercept, na.rm = FALSE, show.legend = NA)
Lline(mapping = NULL, data = NULL, ..., Lintercept, na.rm = FALSE, show.legend = NA)
tline(mapping = NULL, data = NULL, ..., Lintercept, na.rm = FALSE, show.legend = NA)
geom_Rline(mapping = NULL, data = NULL, ..., Rintercept, na.rm = FALSE, show.legend = NA)
Rline(mapping = NULL, data = NULL, ..., Rintercept, na.rm = FALSE, show.legend = NA)
rline(mapping = NULL, data = NULL, ..., Rintercept, na.rm = FALSE, show.legend = NA)
```

Arguments

- `mapping` Set of aesthetic mappings created by `aes()` or `aes_()`.
- `data` The data to be displayed in this layer. There are three options:
  - If `NULL`, the default, the data is inherited from the plot data as specified in the call to `ggplot()`.
  - A `data.frame`, or other object, will override the plot data. All objects will be fortified to produce a data frame. See `fortify()` for which variables will be created.
  - A function will be called with a single argument, the plot data. The return value must be a `data.frame`, and will be used as the layer data. A function can be created from a formula (e.g. `~ head(.x,10)`).
- `...` Other arguments passed on to `layer()`. These are often aesthetics, used to set an aesthetic to a fixed value, like `colour = "red"` or `size = 3`. They may also be parameters to the paired geom/stat.
- `Tintercept, Lintercept, Rintercept` the intercepts for the T, L and R axis respectively
- `na.rm` If `FALSE`, the default, missing values are removed with a warning. If `TRUE`, missing values are silently removed.
show.legend  logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.

Author(s)
Nicholas Hamilton

Examples
ggtern() +
geom_Tline(Tintercept=.5,arrow=arrow(), colour='red') +
geom_Lline(Lintercept=.2, colour='green') +
geom_Rline(Rintercept=.1, colour='blue')

Description
ggplot() initializes a ggplot object. It can be used to declare the input data frame for a graphic and to specify the set of plot aesthetics intended to be common throughout all subsequent layers unless specifically overridden.

Usage
ggplot(data = NULL, mapping = aes(), ..., 
   environment = parent.frame())

## S3 method for class 'ggplot'
print(x, newpage = is.null(vp), vp = NULL, ...)

## S3 method for class 'ggplot'
plot(x, newpage = is.null(vp), vp = NULL, ...)

Arguments
data  Default dataset to use for plot. If not already a data.frame, will be converted to one by fortify(). If not specified, must be supplied in each layer added to the plot.
mapping  Default list of aesthetic mappings to use for plot. If not specified, must be supplied in each layer added to the plot.
...  other arguments not used by this method
environment  DEPRECATED. Used prior to tidy evaluation.
x  plot to display
newpage  draw new (empty) page first?
vp  viewport to draw plot in
Details

`ggplot()` is typically used to construct a plot incrementally, using the + operator to add layers to the existing ggplot object. This is advantageous in that the code is explicit about which layers are added and the order in which they are added. For complex graphics with multiple layers, initialization with `ggplot` is recommended.

There are three common ways to invoke `ggplot`:

- `ggplot(df,aes(x,y,<other aesthetics>))`
- `ggplot(df)`
- `ggplot()`

The first method is recommended if all layers use the same data and the same set of aesthetics, although this method can also be used to add a layer using data from another data frame. See the first example below. The second method specifies the default data frame to use for the plot, but no aesthetics are defined up front. This is useful when one data frame is used predominantly as layers are added, but the aesthetics may vary from one layer to another. The third method initializes a skeleton `ggplot` object which is fleshed out as layers are added. This method is useful when multiple data frames are used to produce different layers, as is often the case in complex graphics.

Value

Invisibly returns the result of `ggplot_build`, which is a list with components that contain the plot itself, the data, information about the scales, panels etc.

Author(s)

Nicholas Hamilton

---

ggsave

Save a ggplot (or other grid object) with sensible defaults (ggtern version)

Description

ggsave() is a convenient function for saving a plot. It defaults to saving the last plot that you displayed, using the size of the current graphics device. It also guesses the type of graphics device from the extension.

Usage

ggsave(filename, plot = last_plot(), device = NULL, path = NULL, scale = 1, width = NA, height = NA, units = c("in", "cm", "mm"), dpi = 300, limitsize = TRUE, ...)

Arguments

- `filename`: File name to create on disk.
- `plot`: Plot to save, defaults to last plot displayed.
- `device`: Device to use (function or any of the recognized extensions, e.g. "pdf"). By default, extracted from filename extension. `ggsave` currently recognises eps/ps, tex (pictex), pdf, jpeg, tiff, png, bmp, svg and wmf (windows only).
- `path`: Path to save plot to (combined with filename).
- `scale`: Multiplicative scaling factor.
- `width`, `height`: Plot dimensions, defaults to size of current graphics device.
- `units`: Units for width and height when specified explicitly (in, cm, or mm)
- `dpi`: Resolution used for raster outputs.
- `limsize`: When `TRUE` (the default), `ggsave` will not save images larger than 50x50 inches, to prevent the common error of specifying dimensions in pixels.
- `...`: Other arguments passed on to graphics device

Author(s)
Nicholas Hamilton

Examples

```r
## Not run:
data(Feldspar)
base = ggtern(Feldspar,aes(Ab,An,Or)) + geom_point()
ggsave("./output.pdf",base,width=10,height=10)
## End(Not run)
```

Description

Plots in `ggtern` are instigated via the default constructor: `ggtern(...)`, which is essentially a convenience wrapper for the following: `ggplot(...) + coord_tern()`, indeed, if one wishes to use `ggplot(...) + coord_tern()` then this is quite satisfactory.

Usage

```r
ggtern(data = NULL, mapping = aes(), ..., 
       environment = parent.frame())
```
Arguments

- **data**: Default dataset to use for plot. If not already a data.frame, will be converted to one by `fortify()`. If not specified, must be supplied in each layer added to the plot.
- **mapping**: Default list of aesthetic mappings to use for plot. If not specified, must be supplied in each layer added to the plot.
- **...**: additional arguments passed through to `ggplot`
- **environment**: DEPRECATED. Used prior to tidy evaluation.

Value

`ggtern(...)` returns an object of class `ggplot`.

Author(s)

Nicholas Hamilton

See Also

For an introduction to the `ggtern` package, (including many examples), click [HERE](#).

Examples

```r
ggtern(data=data.frame(x=1,y=1,z=1),aes(x,y,z)) + geom_point()
```

---

**ggtern_labels**  
*Change Axis labels and legend titles*

Description

New label modification functions, equivalent to the original functions in ggplot2 (`xlab` and `ylab`) however for the new axes used in the `ggtern` package.

Usage

```r
tlab(label, labelarrow = label)
llab(label, labelarrow = label)
rlab(label, labelarrow = label)
wlab(label)
zlab(label)
tarrowlab(label)
```
Arguments

label the desired label
labelarrow the desired label, if different to label, for the markers along the procession arrows

Details

Tlab and xlab are equivalent (when T='x' in the coord_tern definition), as is Llab and ylab (when L='y'), and Rlab and zlab (when R='z'), for other assignments when coord_tern is defined, the equivalence is not the case, however, if T='XXX', then Tlab will be the same as XXXlab (where XXX can be substituted for 'x', 'y' or 'z', and likewise for Llab and Rlab).

zlab is new to ggtern, but is intended to be an analogous to xlab and ylab as per the definitions in ggplot2.

Arrow Label

Tarrowlab, Larrowlab and Rarrowlab permits setting a different label to the apex labels.

Arrow Label Suffix

Wlab changes the ternary arrow suffix (ie atomic percent, weight percent etc) when the ternary arrows are enabled (see theme_showarrows and weight_percent)

Precedence

AAAlab takes precedence over BBBlab (where AAA represents T, L or R and BBB represents x, y or z)

Use of Expressions

Expressions can be used in the labels, in the event that the user wishes to render formula, subscripts or superscripts, see the last example below.

Creation of Aliasses

Aliases exist for Tlab, Llab, Rlab and Wlab, which are tlab, llab, rlab and wlab. These aliases produce an identical result, and are there for convenience (as opposed to having an error thrown) in the event that the user forgets to use an upper-case letter.

Arguments for these functions can be provided as a character or expression, although other values can be inputed (such as, for example, scalar numeric or logical). ggtern also imports the latex2exp package, and these formats can be parsed too.
Author(s)
Nicholas Hamilton

See Also
ggplot2 labs

Examples

data(Feldspar)
plot <- ggtern(data=Feldspar,aes(Ab,An,Or)) + geom_point() +
  xlab("ABC") + ylab("DEF") + zlab("GHI")

#Alternatives, and Arrow Label
plot + Tlab("TOP") + Llab("LHS") + Rlab("RHS") +
  Tarrowlab("Top Arrow Label") + Larrowlab("Left Arrow Label") + Rarrowlab("Right Arrow Label") +
  theme_showarrows() + Wlab("WEIGHT")

#Demonstrate the use of the latex2exp integration, and separate arrow labels.
ggtern(data=Feldspar,aes(x=Ab,y=An,z=Or)) +
  labs( x = "NaAlSi_3O_8",
       xarrow = "Albite, NaAlSi_3O_8",
       y = "(Na,K)AlSi_3O_8",
       yarrow = "Anorthite (Na,K)AlSi_3O_8",
       z = "KAlSi_3O_8",
       zarrow = "Orthoclase KAlSi_3O_8") +
  theme_latex(TRUE) +
  geom_point() +
  theme_showarrows() +
  theme_clockwise() +
  weight_percent()

---

**ggtern_labels_arrow_suffix**

**Atomic, Weight or Custom Percentage Suffix**

Description

By default there are no suffixes behind the arrow label marker (the arrow up next to the ternary axes), and these functions appends to the set of arrow labels, a value to indicate the nature of the scale.

percent_weight adds 'Wt. %' to the arrow marker label as a suffix
weight_percent is an alias for percent_weight()
percent_atomic adds 'At. %' to the arrow marker label as a suffix
atomic_percent is an alias for percent_atomic()
percent_custom adds a custom suffix to the arrow label marker.
custom_percent is an alias for percent_custom()
Usage

percent_weight()
weight_percent()
percent_atomic()
atomic_percent()
percent_custom(x)
custom_percent(x)

Arguments

x the custom suffix

Details

These are convenience wrappers to labs(W="XYZ").

Author(s)

Nicholas Hamilton

See Also

Convenience functions for T,L,R,W labels

Description

Ternary diagrams are used frequently in a number of disciplines to graph compositional features for mixtures of three different elements or compounds. It is possible to represent a coordinate system having three (3) degrees of freedom, in 2D space, since the third dimension is linear and depends only on the other two.

The ggtern package is based on (extends) the very popular ggplot2 package, which is an implementation of Wilkinsons "The Grammar of Graphics", and, makes provision for a highly methodical construction process for the development of meaningful (graphical) data representations. Of course, the above book by Wilkinson outlines the theory, whilst Hadley Wickhams ggplot2 implementation is where much of the magic happens, and, an ideal base-platform for the ggtern package.

In this document, some of the main features are highlighted, however, current examples (and corresponding outputs) can be viewed at http://ggtern.com
ggtern Constructor

Plots in ggtern are instigated via the default constructor: ggtern(...), for additional information, click HERE:

ggtern Ternary Coordinate System

The foundation of this package, is the ternary coordinate system, which can be produced with the coord_tern(...) command and added to an existing ggplot object. The ggtern(...) constructor adds the coord_tern(...) coordinate system by default. For further information on the coord_tern(...) coordinate system, click HERE.

ggtern Valid Geometries

ggplot2, using the grid and proto architectures, makes provision for a many number of geometries to be added progressively in 'layers' to a given base plot. Due to the nature of the ternary coordinate system, some of the geometries which are available in ggplot2, are not relevant (or won’t function) with ternary plots and as such, a limited number of 'approved' geometries can be used. Click HERE for the full list of approved geometries.

Notably, ggtern includes novel geometries not available to ggplot2 which include:

1. Confidence Intervals via the Mahalanobis Distance
2. Ternary Errorbars
3. Ternary Constant-Lines

ggtern Handling Non-Approved Geometries

If a geometric layer is added that is NOT contained in the approved list, IT WILL BE STRIPPED / IGNORED from the ternary diagram when rendering takes place (notifying the user to such effect). The reason for this is that subtle 'patches' have been applied, which are mainly to do with the transformation procedures when incorporating a 'third' dimension. NB: In the future, others may be made available once patched.

ggtern New Theme Elements and Hierarchies

ggtern implements many new theme elements and heirarchies which can be tailored on a case-by-case basis. The full list of new elements can is provided HERE.

ggtern Theme Element Convenience Functions

ggtern has made available a number of convenience functions, for rapid tweaking of common theme elements, for a comprehensive list, see HERE.

ggtern Modification to Required Aesthetics

Each geometry has a pre-determined set of required aesthetics. These have been modifid such that where x and y were previously required, now an additional z aesthetic is required (geom_segment now requires z and zend). This is made possible without affecting the standard ggplot2 behaviour because ggtern distinguishes between ggplot2 and ggtern objects, distinguished by the presence of the coord_tern(...) coordinate system.
**ggtern** **Provided Datasets**

ggtern ships with a number of datasets, including:

1. Elkin and Groves Feldspar Data
2. USDA Textural Classification Data
3. Grantham and Valbel Rock Fragment Data

**Author(s)**

Nicholas Hamilton

**References**

To cite this package, please use the following:


A bibtex entry can be obtained by executing the following command: citation('ggtern')

**Examples**

```r
##-----------------------------------------------
## Basic Usage
##-----------------------------------------------

df = data.frame(x = runif(50),
                 y = runif(50),
                 z = runif(50),
                 Value = runif(50,1,10),
                 Group = as.factor(round(runif(50,1,2))))

ggtern(data=df,aes(x,y,z,color=Group)) +
       theme_rgbw() +
       geom_point() + geom_path() +
       labs(x="X",y="Y",z="Z",title="Title")
```

**Description**

Themes set the general aspect of the plot such as the colour of the background, gridlines, the size and colour of fonts.
Usage
theme_ggtern(base_size = 11, base_family = "")
theme_gray(base_size = 11, base_family = "")
theme_bw(base_size = 12, base_family = "")
theme_linedraw(base_size = 12, base_family = "")
theme_light(base_size = 12, base_family = "")
theme_minimal(base_size = 12, base_family = "")
theme_classic(base_size = 12, base_family = "")
theme_dark(base_size = 12, base_family = "")
theme_void(base_size = 12, base_family = "")
theme_darker(base_size = 12, base_family = ")
theme_custom(base_size = 12, base_family = "",
  tern.plot.background = NULL, tern.panel.background = NULL,
  col.T = "black", col.L = "black", col.R = "black",
  col.grid.minor = "white")
theme_rgbw(base_size = 12, base_family = "")
theme_rgbg(base_size = 12, base_family = "")
theme_matrix(base_size = 12, base_family = "")
theme_tropical(base_size = 12, base_family = "")
theme_bluedark(base_size = 12, base_family = "")
theme_bluelight(base_size = 12, base_family = ")
theme_bvbw(base_size = 12, base_family = "")
theme_bvbg(base_size = 12, base_family = "")

Arguments
  base_size        base font size
  base_family      base font family
  tern.plot.background
                  colour of background colour to plot area
tern.panel.background  
   colour of panel background of plot area

col.T  
colour of top axis, ticks labels and major gridlines

col.L  
colour of left axis, ticks, labels and major gridlines

col.R  
colour of right axis, ticks, labels and major gridlines

col.grid.minor  
the colour of the minor grid theme_custom is a convenience function to allow the user to control the basic theme colours very easily.

**Details**

**theme_gray** The signature ggplot2 theme with a grey background and white gridlines, designed to put the data forward yet make comparisons easy.

**theme_bw** The classic dark-on-light ggplot2 theme. May work better for presentations displayed with a projector.

**theme_linedraw** A theme with only black lines of various widths on white backgrounds, reminiscent of a line drawings. Serves a purpose similar to theme_bw. Note that this theme has some very thin lines (< 1 pt) which some journals may refuse.

**theme_light** A theme similar to theme_linedraw but with light grey lines and axes, to direct more attention towards the data.

**theme_dark** The dark cousin of theme_light, with similar line sizes but a dark background. Useful to make thin coloured lines pop out.

**theme_darker** A darker cousin to theme_dark, with a dark panel background.

**theme_minimal** A minimalistic theme with no background annotations.

**theme_classic** A classic-looking theme, with x and y axis lines and no gridlines.

**theme_rgbw** A theme with white background, red, green and blue axes and gridlines

**theme_rgbg** A theme with grey background, red, green and blue axes and gridlines

**theme_void** A completely empty theme.

**theme_custom** Theme with custom basic colours

**theme_matrix** Theme with very dark background and bright green features

**theme_tropical** Theme with tropical colours

**theme_bluelight** A blue theme with light background and dark features

**theme_bluedark** A blue theme with dark background and light features

**theme_bvbw** A black/vermillion/blue theme with white background, for colorblind sensitive readers, see references.

**theme_bvbg** A black/vermillion/blue theme with grey background, for colorblind sensitive readers, see references.

**Author(s)**

Nicholas Hamilton
References

Okabe, Masataka, and Kei Ito. "How to make figures and presentations that are friendly to color blind people." University of Tokyo (2002). http://jfly.iam.u-tokyo.ac.jp/color/

Examples

# Create a list of the theme suffixes
themesOrg = c('gray', 'bw', 'linedraw', 'light',
               'dark', 'minimal', 'classic', 'void')
themesNew = c('custom', 'darker', 'rgbw', 'rgbg', 'tropical',
              'matrix', 'bluelight', 'bluedark', 'bvbw', 'bvbg')

# Iterate over all the suffixes, creating a list of plots
plotThemes = function(themes){
  grobs = lapply(themes, function(x){
    thmName = sprintf("theme_%s", x)
    thm = do.call(thmName, args = list(base_size = 9))
    df = data.frame(label = thmName)
    ggtern(df) + facet_wrap(~label) + thm
  })
  grobs
}

# Arrange the Original Themes
grid.arrange(grobs = plotThemes(themesOrg), top = "Collection of Themes (Original)"

# Arrange the New Themes
grid.arrange(grobs = plotThemes(themesNew), top = "Collection of Themes (New Themes)"

---

## labels_tern Generate Axis Labels

### Description

Calculates the Labels for Major or Minor Gridlines based on the input limits.

### Usage

labels_tern(limits = c(0, 1), breaks = breaks_tern(limits),
            format = "%g", factor = 100)

### Arguments

- **limits**: the scale limits
- **breaks**: numeric denoting the breaks to produce corresponding labels
- **format**: the formatting string to be passed through to the `sprintf` function
- **factor**: the multiplicative factor
**Author(s)**

Nicholas Hamilton

**Examples**

```r
labels_tern()
labels_tern(limits = c(0,.5))
```

---

**Description**

`label_formatter` is a function that formats / parses labels for use in the grid.

**Usage**

```r
label_formatter(label, ...)
```

**Arguments**

- `label` character label
- `...` additional arguments

---

**mahalanobis_distance**  

**Mahalanobis Distance**

**Description**

Modified version of the code provided in the `drawMahal` package

**Usage**

```r
mahalanobis_distance(x, x.mean, x.cov, whichlines = c(0.975, 0.9, 0.75), m = 360)
```

**Arguments**

- `x` data
- `x.mean` mean value
- `x.cov` covariance value
- `whichlines` the confidence values
- `m` the number of values to return for each line
Value

list containing mdX and mdY values.

Author(s)

Nicholas Hamilton

---

position_jitter_tern  Jitter Ternary Points

Description

Jitter ternary points to avoid overplotting.

Usage

position_jitter_tern(x = NULL, y = NULL, z = NULL)

Arguments

x, y, z  amount of positional jitter

Author(s)

Nicholas Hamilton

See Also

Other position adjustments: position_nudge_tern

---

position_nudge_tern  Nudge Ternary Points.

Description

This is useful if you want to nudge labels a little ways from their points, input data will normalised to sum to unity before applying the particular nudge, so the nudge variables should be as a fraction ie (0,1)

Usage

position_nudge_tern(x = 0, y = 0, z = 0)

Arguments

x, y, z  Amount of compositions to nudge
**predictdf2d**

**Prediction data frame**

**Description**

Get predictions with standard errors into data frame

**Usage**

`predictdf2d(model, xseq, yseq)`

**Arguments**

- `model`: the model to predict
- `xseq, yseq`: the x and y values

**scale_X_continuous**

**Ternary Position Scales**

**Description**

Define the ternary continuous position scales (T, L & R).

**Usage**

- `scale_T_continuous(name = waiver(), limits = NULL, breaks = waiver(), minor_breaks = waiver(), labels = waiver(), ...)`
- `scale_L_continuous(name = waiver(), limits = NULL, breaks = waiver(), minor_breaks = waiver(), labels = waiver(), ...)`
- `scale_R_continuous(name = waiver(), limits = NULL, breaks = waiver(), minor_breaks = waiver(), labels = waiver(), ...)`
Arguments

name
The name of the scale. Used as the axis or legend title. If waiver(), the default, the name of the scale is taken from the first mapping used for that aesthetic. If NULL, the legend title will be omitted.

limits
One of:
- NULL to use the default scale range
- A numeric vector of length two providing limits of the scale. Use NA to refer to the existing minimum or maximum
- A function that accepts the existing (automatic) limits and returns new limits Note that setting limits on positional scales will remove data outside of the limits. If the purpose is to zoom, use the limit argument in the coordinate system (see coord_cartesian()).

breaks
One of:
- NULL for no breaks
- waiver() for the default breaks computed by the transformation object
- A numeric vector of positions
- A function that takes the limits as input and returns breaks as output (e.g., a function returned by scales::extended_breaks())

minor_breaks
One of:
- NULL for no minor breaks
- waiver() for the default breaks (one minor break between each major break)
- A numeric vector of positions
- A function that given the limits returns a vector of minor breaks.

labels
One of:
- NULL for no labels
- waiver() for the default labels computed by the transformation object
- A character vector giving labels (must be same length as breaks)
- A function that takes the breaks as input and returns labels as output

... not used

Author(s)
Nicholas Hamilton

strip_unapproved
Strip Unapproved Layers

Description

strip_unapproved is an internal function which essentially 'deletes' layers from the current ternary plot in the event that such layers are not one of the approved layers. For a layer to be approved, it must use an approved geometry, and also an approved stat. Refer to approved_layers for the current list of approved geometries and stats.
ternary_transformation

Usage

strip_unapproved(layers)

Arguments

layers list of the layers to strip unapproved layers from.

Value

strip_unapproved returns a list of approved layers (may be empty if none are approved).

ternary_transformation

Ternary / Cartesian Transformation

Description

Functions to transform data from the ternary to cartesian spaces and vice-versa.

Usage

tlr2xy(data, coord, ..., inverse = FALSE, scale = TRUE, drop = FALSE)
xy2tlr(data, coord, ..., inverse = FALSE, scale = TRUE)

Arguments

data data.frame containing columns as required by the coordinate system. Data will be scaled so that the rows sum to unity, in the event that the user has provided data that does not.
coord Coordinate system object, inheriting the CoordTern class, error will be thrown if a different coordinate system is sent to this method
... not used
inverse logical if we are doing a forward (FALSE) or reverse (TRUE) transformation
scale logical as to whether the transformed coordinates are scaled (or reverse scaled in the case of inverse transformation) according to the training routine defined in the coordinate system.
drop drop all non columns which are not involved in the transformation

Details

tlr2xy transforms from the ternary to cartesian spaces, an inverse transformation transforms between cartesian to ternary spaces
xy2tlr transforms from the cartesian to ternary spaces, an inverse transformation transforms between ternary to cartesian spaces, it is the reciprocal to tlr2xy, therefore an inverse transformation in xy2tlr function is the same as the forward transformation in tlr2xy
tern_limits

Author(s)
Nicholas Hamilton

Examples

data(Feldspar)
dfm = plyr::rename(Feldspar,c("Ab"="x","An"="y","Or"="z"))
crd = coord_tern()
fwd = tlr2xy(dfm,crd)
rev = tlr2xy(fwd,crd,inverse = TRUE)

tern_limits

Restrict Ternary Limits

Description

tern_limits (or its aliases) appends new T, L and R ternary continuous scales, where the maximum scale value is specified, and, where the minimums for each are solved.

Usage

tern_limit(T = 1, L = 1, R = 1, ...)

limit_tern(...)  

Arguments

T, L, R   numeric value (scalar) of the maximum T, L, R species limit for each scale respectively
...
other arguments to pass to ALL of scale_X_continuous (X = T, L, R)

Details

The contra value (ie minimum value) for the T, L and R species is solved using linear equations, therefore, if the solution is degenerate, or, the solution results in a zero range in either of the proposed scales, then a warning message will be reported and an empty list returned. Note that limits_tern(...), limit_tern(...) and tern_limit(...) are all aliases for the main function, tern_limits(...) and can be used interchangeably.

Value

Either an empty list (when no solution can be found), or a list containing one of each of scale_X_continuous (X = T, L, R)

Author(s)
Nicholas Hamilton
See Also

scale_T_continuous, scale_L_continuous and scale_R_continuous

Examples

#Display a non-zoomed and zoomed plot side by side
data(Feldspar)
df.lims = data.frame(Ab = c(1,.25,.25),
                     An = c(0,.75,.00),
                     Or = c(0,.00,.75))

#Build the non-zoomed plot
A = ggtern(Feldspar,aes(Ab,An,Or)) +
  stat_density_tern(geom='polygon',aes(fill=..level..,alpha=..level..)) +
  geom_point() +
  geom_mask() +
  geom_polygon(data=df.lims,color='red',alpha=0,size=0.5) +
  guides(color='none',fill='none',alpha='none') +
  labs(title = "Non-Zoomed")

#Build the zoomed plot
B = A +
  tern_limits(T=max(df.lims$An), L=max(df.lims$Ab), R=max(df.lims$Or)) +
  labs(title = "Zoomed")

#Arrange the above plots side by side for illustration
grid.arrange(A,B,ncol=2,top="Demonstration of Limiting Region")

theme_arrowlength  Change the Length of the Ternary Arrows

Description

A set of convenience functions to rapidly change the length of the ternary arrows, the convenience
functions include presets (short, normal, long), or makes provision for the user to specify custom
fractional starting and ending values relative to the size of the ternary axis. In the event that the
user elects to specify the values via the theme_arrowcustomlength (or its aliasses), then the user
can specify a single scalar value which apply to all three (3) arrows, or, alternatively, can provide a
numeric vector of length three (3), one for each arrow respectively.

Usage

theme_arrowcustomlength(start = getOption("tern.arrow.start"),
                         finish = getOption("tern.arrow.finish"))

theme_arrowlength(start = getOption("tern.arrow.start"),
                   finish = getOption("tern.arrow.finish"))

theme_arrowsmall()
theme_arrowsmall() or theme_arrowshort() reduces the ternary arrows to short arrows, occupying a length between 0.4 and 0.6 of the length of the ternary axis.

Normal/Default Arrow Length

theme_arrownormal() or theme_arrowdefault() reduces the ternary arrows to normally sized arrows, occupying a length between `getOption("tern.arrow.start")` and `getOption("tern.arrow.finish")` global option values, whatever they may be.

Long Arrow Length

theme_arrowlarge() or theme_arrowlong() increases the ternary arrows to long arrows occupying a length between 0.2 and 0.8 of the length of the ternary axis.
theme_bordersontop

Author(s)

Nicholas Hamilton

See Also

theme_arrowbaseline and theme(tern.axis.arrow.sep=X) for methods to adjust the separation distance of the ternary arrows from the ternary axes.

Examples

#Create base plot
plot <- ggtern(data=data.frame(x=1,y=1,z=1),aes(x,y,z)) + geom_point()

#Pre-Specified Values
plot + theme_arrowsmall()

## Alternatives, Uncomment lines below
plot + theme_arrornormal()
plot + theme_arrowlarge()
plot + theme_arrowcustomlength(.1,.8)
plot + theme_arrowlength(start=c(.1,.25,.4),finish=c(.9,.75,.6))

theme_bordersontop       Render Borders on Top

Description

Convenience functions to render the axis border lines on top (or bottom) of the other layers. By default the borders are rendered in the background (bottom)

Usage

theme_bordersontop()

theme_bordersonbottom()

Author(s)

Nicholas Hamilton
theme_complete

---

theme_clockwise  Direction of Ternary Rotation

Description

theme_clockwise, theme_anticlockwise (or their aliases) are functions that instruct the axes precession to be clockwise or anticlockwise respectively.

Usage

theme_clockwise()

theme_anticlockwise()

theme_counterclockwise()

Details

If the `tern.axis.arrow.show` value is FALSE, these functions will set it to TRUE.

Author(s)

Nicholas Hamilton

---

theme_complete  List of Available Themes

Description

ggtern ships with a number of complete themes, summarized as follows. These themes combine the base themes available to ggplot2 and a number of NEW themes, which are unique to ggtern.

- Black and White Theme: `theme_bw()`
- Minimal Theme: `theme_minimal()`
- Classic Theme: `theme_classic()`
- Gray and White Theme: `theme_gray()`
- Red, Green, Blue and White Theme: `theme_rgbw()`
- Red, Green, Blue and Gray Theme: `theme_rgbg()`
- Dark Theme: `theme_dark()`
- Darker Theme: `theme_darker()`
- Light Theme: `theme_light()`
- Theme with Only Black Lines: `theme_linedraw()`
- Matrix Theme: `theme_matrix()`
• Tropical Theme: \texttt{theme_tropical(...)}
• BlueLight Theme: \texttt{theme_bluelight(...)}
• BlueDark Theme: \texttt{theme_bluedark(...)}
• Black Vermillion Blue Theme (White Background): \texttt{theme_bvbw(...)}
• Black Vermillion Blue Theme (Grey Background): \texttt{theme_bvbg(...)}

\textbf{Author(s)}
Nicholas Hamilton

\textbf{See Also}
\texttt{ggtern\_themes}

\textbf{Description}
\texttt{ggtern} has made available a number of convenience functions for rapid tweaking of the various theme elements, for a full list of the available theme elements which can be manually modified, see \texttt{HERE}.

\textbf{Convenience Functions}
Some of the Convenience functions that ship with \texttt{ggtern}, to assist in the rapid modification of key theme elements:

- Show/Hide Axis Titles
- Show/Hide Arrows
- Show/Hide Grids
- Show/Hide Axis Ticklabels
- Show/Hide Primary/Secondary Ticks
- Ticks Inside or Outside of the Main Plot Area
- Set Length of arrows
- Clockwise/Anticlockwise Axis Precession
- Rotate the plot by X degrees or radians
- Create a mesh of 'n' Major/Minor gridlines
- Enable/Disable parsing of labels according to latex markup
- Turn off the clipping mask
- Atomic or Weight Percent Arrow Label Suffix.
Manual Modification

For manual modification on a per-element basis:

- **Ternary Theme Elements**

Default Themes

Default (complete) themes which ship with ggtern:

- **Complete Themes**

Examples

```r
# Load data and create the base plot.
plot <- ggtern() + theme_bw() +
  theme(tern.axis.ticks.length.major=unit(3.0, 'mm'),
        tern.axis.ticks.length.minor=unit(1.5, 'mm'))
plot

# Show Arrows
last_plot() + theme_showarrows()

# Major/Minor Grids?
last_plot() + theme_nogrid_minor()
last_plot() + theme_nogrid_major()
last_plot() + theme_showgrid()

# Clockwise/Anticlockwise Precession
last_plot() + theme_clockwise()

# Ticks Inside or Outside
last_plot() + theme_ticksinside()

# Show/Hide BOTH Primary and Secondary Ticks
last_plot() + theme_showticks()
last_plot() + theme_hideticks()

# Show/Hide EITHER Primary OR Secondary Ticks.
last_plot() + theme_showprimary() + theme_hidesecondary()
last_plot() + theme_hideprimary() + theme_showsecondary()

# Atomic / Weight Percent
last_plot() + theme_showarrows() + atomic_percent() # + weight_percent()
last_plot() + theme_showarrows() + custom_percent("Atomic Percent")

# Rotation
last_plot() + theme_rotate(60)
```
Description

`ggtern` creates many new theme elements and inheritances, the following is an outline:

Details

Theme elements can inherit properties from other theme elements. For example, `axis.title.x` inherits from `axis.title`, which in turn inherits from `text`. All text elements inherit directly or indirectly from `text`; all lines inherit from `line`, and all rectangular objects inherit from `rect`.

Modifying the newly created items requires the same procedures as introduced in the `ggplot2 theme` documentation. Some convenience functions have been also newly created, proceed to `theme_convenience_functions` for additional information.

New/Additional Inheritance Structures

Based on the `ggplot2` existing structure (`theme`), the **NEW** individual theme elements for the ternary plot are as follows:

<table>
<thead>
<tr>
<th>NAME</th>
<th>OBJECT/(INHERITS)</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>line</td>
<td>element_line</td>
<td></td>
</tr>
<tr>
<td>rect</td>
<td>element_rect</td>
<td></td>
</tr>
<tr>
<td>text</td>
<td>element_text</td>
<td></td>
</tr>
<tr>
<td>title</td>
<td>element_text/(text)</td>
<td></td>
</tr>
<tr>
<td>axis.line</td>
<td>element_line/(line)</td>
<td></td>
</tr>
<tr>
<td>axis.text</td>
<td>element_text/(text)</td>
<td></td>
</tr>
<tr>
<td>axis.title</td>
<td>element_text/(title)</td>
<td></td>
</tr>
<tr>
<td>axis.ticks</td>
<td>element_line/(line)</td>
<td></td>
</tr>
<tr>
<td>legend.key.size</td>
<td>unit</td>
<td></td>
</tr>
<tr>
<td>panel.grid</td>
<td>element_line/(line)</td>
<td></td>
</tr>
<tr>
<td>panel.grid.major</td>
<td>element_line/(panel.grid)</td>
<td></td>
</tr>
<tr>
<td>panel.grid.minor</td>
<td>element_line/(panel.grid)</td>
<td></td>
</tr>
<tr>
<td>strip.text</td>
<td>element_text/(text)</td>
<td></td>
</tr>
<tr>
<td>axis.line.x</td>
<td>element_line/(axis.line)</td>
<td></td>
</tr>
<tr>
<td>axis.line.x.top</td>
<td>element_line/(axis.line.x)</td>
<td></td>
</tr>
<tr>
<td>axis.line.x.bottom</td>
<td>element_line/(axis.line.x)</td>
<td></td>
</tr>
<tr>
<td>axis.line.y</td>
<td>element_line/(axis.line)</td>
<td></td>
</tr>
<tr>
<td>axis.line.y.left</td>
<td>element_line/(axis.line.y)</td>
<td></td>
</tr>
<tr>
<td>axis.line.y.right</td>
<td>element_line/(axis.line.y)</td>
<td></td>
</tr>
<tr>
<td>axis.text.x</td>
<td>element_text/(axis.text)</td>
<td></td>
</tr>
<tr>
<td>axis.text.x.top</td>
<td>element_text/(axis.text.x)</td>
<td></td>
</tr>
<tr>
<td>axis.text.x.bottom</td>
<td>element_text/(axis.text.x)</td>
<td></td>
</tr>
<tr>
<td>axis.text.y</td>
<td>element_text/(axis.text)</td>
<td></td>
</tr>
<tr>
<td>axis.text.y.left</td>
<td>element_text/(axis.text.y)</td>
<td></td>
</tr>
<tr>
<td>axis.text.y.right</td>
<td>element_text/(axis.text.y)</td>
<td></td>
</tr>
</tbody>
</table>
axis.ticks.length  unit
axis.ticks.length.x  unit/(axis.ticks.length)
axis.ticks.length.x.top  unit/(axis.ticks.length.x)
axis.ticks.length.x.bottom  unit/(axis.ticks.length.x)
axis.ticks.length.y  unit/(axis.ticks.length)
axis.ticks.length.y.left  unit/(axis.ticks.length.y)
axis.ticks.length.y.right  unit/(axis.ticks.length.y)
axis.ticks.x  element_line/(axis.ticks)
axis.ticks.x.top  element_line/(axis.ticks.x)
axis.ticks.x.bottom  element_line/(axis.ticks.x)
axis.ticks.y  element_line/(axis.ticks)
axis.ticks.y.left  element_line/(axis.ticks.y)
axis.ticks.y.right  element_line/(axis.ticks.y)
axis.title.x  element_text/(axis.title)
axis.title.x.top  element_text/(axis.title.x)
axis.title.x.bottom  element_text/(axis.title.x)
axis.title.y  element_text/(axis.title)
axis.title.y.left  element_text/(axis.title.y)
axis.title.y.right  element_text/(axis.title.y)
legend.background  element_rect/(rect)
legend.margin  margin
legend.spacing  unit
legend.spacing.x  unit/(legend.spacing)
legend.spacing.y  unit/(legend.spacing)
legend.key  element_rect/(rect)
legend.key.height  unit/(legend.key.size)
legend.key.width  unit/(legend.key.size)
legend.text  element_text/(text)
legend.text.align  character
legend.title  element_text/(title)
legend.title.align  character
legend.position  character
legend.direction  character
legend.justification  character
legend.box  character
legend.box.just  character
legend.box.margin  margin
legend.box.background  element_rect/(rect)
legend.box.spacing  unit
panel.background  element_rect/(rect)
panel.border  element_rect/(rect)
panel.spacing  unit
panel.spacing.x  unit/(panel.spacing)
panel.spacing.y  unit/(panel.spacing)
panel.grid.major.x  element_line/(panel.grid.major)
panel.grid.major.y  element_line/(panel.grid.major)
panel.grid.minor.x  element_line/(panel.grid.minor)
panel.grid.minor.y  element_line/(panel.grid.minor)
theme_gridsontop

panel.onstop logical
strip.background element_rect(rect)
strip.background.x element_rect(strip.background)
strip.background.y element_rect(strip.background)
strip.text.x element_text(strip.text)
strip.text.x.top element_text(strip.text.x)
strip.text.x.bottom element_text(strip.text.x)
strip.text.y element_text(strip.text)
strip.text.y.left element_text(strip.text.y)
strip.text.y.right element_text(strip.text.y)
strip.placement character
strip.placement.x character(strip.placement)
strip.placement.y character(strip.placement)
strip.switch.pad.grid unit
strip.switch.pad.wrap unit
plot.background element_rect(rect)
plot.title element_text(title)
plot.title.position character
plot.subtitle element_text(title)
plot.caption element_text(title)
plot.caption.position character
plot.tag element_text(title)
plot.tag.position character
plot.margin margin
aspect.ratio character

**NB:** tern.panel.background, whilst the ternary area is 'triangular' per-se, element_rect has been used, as it actually holds NO information regarding the geometry (width, height), only fill, color, size and linetype border (ie the style of how it will be rendered).

**Author(s)**

Nicholas Hamilton

<table>
<thead>
<tr>
<th>theme_gridsontop</th>
<th>Render Grids on Top</th>
</tr>
</thead>
</table>

**Description**

Convenience function to render the major and minor grids on top (or bottom) of the other layers. By default the grids are rendered in the background (bottom)

**Usage**

theme_gridsontop()

theme_gridsonbottom()
Description
A series of convenience functions that either enable or disable the use of the \texttt{latex2exp} package for parsing the various text elements using the TeX method. In many cases, by turning the latex parsing on, this prevents confusing use of expressions to obtain greeks, superscripts, subscripts etc... Note that when latex parsing is enabled, this can override specific formatting directives from the element tree, see the third and fourth example below.

Usage

\begin{verbatim}
theme_latex(value = TRUE)
theme_showlatex()
theme_nolatex()
theme_hidelatex()
\end{verbatim}

Arguments

\begin{verbatim}
value logical as to whether to enable latex parsing or not
\end{verbatim}

Author(s)
Nicholas Hamilton

See Also
TeX

Examples

\begin{verbatim}
#Demonstrate without latex parsing
ggtern() +
  theme_latex(FALSE) +
  labs(title = '\textit{Plot Title}')

#Same as before, but turn on the latex parsing
last_plot() +
  theme_latex(TRUE)
\end{verbatim}
# Demonstrate latex overriding the bold face
```r
ggtern() +
  labs(title = '\textit{Plot Title}') +
  theme_latex(TRUE) +
  theme('plot.title' = element_text(face='bold'))
```
# Turn off latex parsing, bold title revealed
```r
last_plot() +
  theme_latex(FALSE)
```

---

**theme_legend_position**  
*Position Legend in Convenient Locations*

**Description**

A convenience function to position the legend at various internal positions

**Usage**
```r
theme_legend_position(x = "topleft")
```

**Arguments**

- `x`  
  the position, valid values are topleft, middleleft, bottomleft, topright, middleright and bottomright, or the shortened versions respectively, tl, ml, bl, tr, mr, br

**Author(s)**

Nicholas Hamilton

---

**theme_mesh**  
*Create Grid Mesh*

**Description**

Convenience function for creation of a grid mesh of an ideal number of 'n' major breaks. Note that the value of 'n' is the target number of breaks, and due to the use of the pretty function within `breaks_tern` convenience function, may not be strictly adhered or reflected.

**Usage**
```r
theme_mesh(n = 5, ...)
```

**Arguments**

- `n`  
  the 'target' number of major breaks
- `...`  
  additional arguments to be passed through to `tern_limits`
Author(s)
Nicholas Hamilton

Examples

# Default example of a target n=10 mesh
ggtern() +
  theme_mesh(10)

# Default example, of a target n=5 mesh, with limiting region
ggtern() +
  theme_mesh(5, T=.5, L=.5, R=.5)

theme_noarrows Show or Hide the Ternary Arrows

Description

theme_noarrows is a function that appends to the current theme a flag to switch OFF the ternary arrows

Usage

theme_noarrows()

theme_hidearrows()

theme_showarrows()

Author(s)
Nicholas Hamilton

theme_nomask Show or Hide the Clipping Mask

Description

Convenience Function to Show or Hide the Clipping Mask, theme_showmask is a function that appends to the current theme a flag to switch ON the clipping mask, whilst, theme_nomask (or theme_hidemask) is a function that appends to the current theme a flag to switch OFF the clipping mask
theme_novar_tern

Usage

theme_nomask()
theme_hidemask()
theme_showmask()

Author(s)

Nicholas Hamilton

theme_novar_tern  
Blank one variable's annotations in ternary plot

Description

This function blanks the grid and axis elements for one variable in a ternary plot.

Usage

theme_novar_tern(species, ...)

Arguments

species  
A character giving the species. Choices are "T," "L," and "R," but is not case sensitive

...  
Further arguments, including additional selections otherwise used in species

Details

This function takes a user-specified character corresponding to one of the three ternary variables, and constructs a theme function which adds blank elements for that variable’s grid elements and axis elements chosen from the ggtern package. This new function is then executed which "adds" this theme to the open ternary plot.

The logic of the species selection is pretty transparent so it may be possible to customize this function to add further affected elements as desired. However the computing on the language which drives this function has not been thoroughly tested. Neither has this function been tested with non-ternary plots available in the ggplot2 framework.

Value

This function is called for the side effect of adding a theme which actually blanks the grid and axis elements for the chosen ternary species.

Author(s)

Nicholas Hamilton, John Szumiloski
Examples

```r
base = ggtern() + theme_rgbg()
base + theme_novar_tern("L")
base + theme_novar_tern(c("T","L"))
base + theme_novar_tern('L',R)
```

---

**theme_rotate**  
*Rotate Ternary Diagram*

Description

Convenience function to rotate the diagram by an angle in degrees or radians.

Usage

```r
theme_rotate(degrees = 60, radians = degrees * pi/180)
```

Arguments

- `degrees`, `radians`
  
  specify the angle to rotate the plot by in either degrees or radians. If both degrees and radians are specified, then precedence is given to the `radians` argument. If no value is specified, the plot will rotate by 60 degrees

Author(s)

Nicholas Hamilton

Examples

```r
x = ggtern(data.frame(x=1,y=1,z=1),aes(x,y,z))
for(a in seq(0,60,by=15))
  print(x + theme_rotate(a))
```

---

**theme_showgrid**  
*Show or Hide Grid*

Description

A set of convenience functions to enable or disable the use of major or minor (or both) gridlines.
Usage

theme_showgrid()
theme_hidegrid()
theme_nogrid()
theme_tern_nogrid()
theme_showgrid_major()
theme_hidegrid_major()
theme_nogrid_major()
theme_tern_nogrid_major()
theme_showgrid_minor()
theme_hidegrid_minor()

Details

These flags operate at the 'rendering' level, and, supercede the presence of theme elements, therefore,
theme_hidegrid(...) or its aliases will PREVENT rendering of grid elements, irrespective of whether those grid elements are valid (renderable). From the counter perspective,
theme_showgrid(...) or its aliases will ALLOW rendering of grid elements, subject to those grid elements being valid (renderable, i.e. say element_line as opposed to element_blank).

theme_hidegrid or theme_nogrid (alias) is a function which disables both MAJOR and MINOR gridlines.

theme_showgrid_major is a function which enables MAJOR gridlines.

theme_hidegrid_major or theme_nogrid_major (alias) is a function which disables MAJOR gridlines.

theme_showgrid_major is a function which enables MINOR gridlines.

theme_hidegrid_minor or theme_nogrid_minor (alias) is a function which disables MINOR gridlines.

theme_showgrid is a function which enables both MAJOR and MINOR gridlines.

Author(s)

Nicholas Hamilton

Examples

# Load data
data(Feldspar)
plot <- ggtern(data=Feldspar, aes(Ab, An, Or)) +
    geom_point() + #Layer
    theme_bw() #For clarity
plot
plot = plot + theme_hidegrid(); plot
plot + theme_showgrid()

---

### theme_showlablels

**Show or Hide Axis Ticklabels**

**Description**

Convenience functions to enable or disable the axis ticklabels

**Usage**

```r
theme_showlablels()
theme_hidelablels()
theme_nolablels()
```

**Details**

`theme_showlablels` is a function that appends to the current theme a flag to switch ON the axis ticklabels, whilst `theme_hidelablels` or `theme_nolablels` (Alias) are functions that appends to the current theme a flag to switch OFF the axis ticklabels.

**Author(s)**

Nicholas Hamilton

---

### theme_showprimary

**Show or Hide the Primary/Secondary Ticks**

**Description**

Convenience functions to enable or disable the axis primary or secondary ticks.
Usage

theme_noprimary()
theme_hideprimary()
theme_showprimary()
theme_nosecondary()
theme_hidesecondary()
theme_showsecondary()
theme_showticks()
theme_hideticks()
theme_noticks()

Details

In ggtern, the primary ticks are deemed as being the ticks along the binary axis increasing to the apex species, primary ticks can consist of both major and minor ticks (major ticks have labels, and are generally longer and bolder). Therefore, there are three (3) sets of major primary ticks, and, three (3) sets of minor primary ticks.

These convenience functions introduce the concept of secondary ticks, which, are the same items however on the 'opposing' binary axis.

For example, considering the TOP apex species, in a plot with 'clockwise' axis precession, the primary ticks would run along the LHS, whilst, the secondary ticks, would run along the RHS. By default, the primary ticks are switched ON, whilst the secondary ticks are switched OFF and are controlled by the `tern.axis.ticks.primary.show` and `tern.axis.ticks.secondary.show` theme elements respectively.

theme_showsecondary is a function that apends to the current theme a flag to switch ON the secondary ticks `theme_showticks()`, `theme_hideticks()`, `theme_noticks()` are functions that switch ON or OFF BOTH the primary or secondary ticks. `theme_nosecondary` or `theme_hidesecondary` (Alias) are functions that apends to the current theme a flag to switch OFF the secondary ticks `theme_showprimary` is a function that apends to the current theme a flag to switch ON the primary ticks `theme_noprimary` or `theme_hideprimary` (Alias) are functions that apends to the current theme a flag to switch OFF the primary ticks

Author(s)

Nicholas Hamilton

Examples

data(Feldspar)
plot <- ggtern(data=Feldspar,aes(Ab,An,Or)) + geom_point() +
### theme_ticklength

**Description**

Convenience Function for changing the major and/or minor ticklengths.

**Usage**

```r
theme_ticklength(major = NULL, minor = NULL)
theme_ticklength_major(major)
theme_ticklength_minor(minor)
```

**Arguments**

- `major`, `minor`  
  length of major and minor ticklengths respectively. Must be a unit object, or will be ignored.

---

### theme_showtitles

*Show or Hide the Axis (Apex) Titles*

**Description**

Convenience functions to SHOW or HIDE the apex labels.

**Usage**

```r
theme_showtitles()
theme_hidetitles()
theme_notitles()
```
theme_ticksoutside

Author(s)

Nicholas Hamilton

Examples

ggtern() +
theme_ticklength(major = unit(5.0, 'mm'),
minor = unit(2.5, 'mm'))

theme_ticksoutside  Place Ticks Inside or Outside

Description

theme_ticksoutside is a function that ensures the ticks are placed OUTSIDE of the plot area, whereas, theme_ticksinside is a function that ensures the ticks are placed INSIDE of the plot area (opposite to theme_ticksoutside)

Usage

theme_ticksoutside()

theme_ticksinside()

Author(s)

Nicholas Hamilton

theme_zoom_X  Zoom on Plot Region

Description

A series of convenience functions for the zooming in on the middle or apex regions to various degrees. In these convenience functions, a single value of x is expected, which defines the values of the apex limits other than the point of reference, for example, theme_zoom_T will fix the T limit at 1, and will adjust the balancing limits according to the argument x. Equivalent are also possible for the L and R apexes, via the theme_zoom_L and theme_zoom_R functions respectively. Finally, the theme_zoom_center function will adjust all three apex limits, serving, as the name suggests, to act as a centred zoom. The examples below are fairly self explanatory.
**theme_zoom_X**

**Usage**

theme_zoom_T(x = 1, ...)

theme_zoom_L(x = 1, ...)

theme_zoom_R(x = 1, ...)

theme_zoom_center(x = 1, ...)

**Arguments**

x numeric scalar

... additional arguments to be passed through to limit_tern

**Author(s)**

Nicholas Hamilton

**Examples**

#Default Plot
data(Feldspar)
base = ggtern(Feldspar,aes(Ab,An,Or)) +
    theme_bw(8) +
    geom_density_tern() +
    geom_point() +
    labs(title="Original")

#Zoom on Left Region
A = base + theme_zoom_L(0.5) + labs(title="theme_zoom_L")

#Zoom on Right Region
B = base + theme_zoom_R(0.5) + labs(title="theme_zoom_R")

#Zoom on Top Region
C = base + theme_zoom_T(0.5) + labs(title="theme_zoom_T")

#Zoom on Center Region
D = base + theme_zoom_center(0.5) + labs(title="theme_zoom_center")

#Put all together for comparisons sake
grid.arrange(arrangeGrob(base),
    arrangeGrob(A,B,nrow=1),
    arrangeGrob(C,D,nrow=1),
    ncol=1, heights=c(2,1,1),
    top = "Comparison of Zooming Functions")
Depreciated Functions

Description

The following is a list of functions which were once used in previous versions of ggtern, however, have now been depreciated

**DEPRECIATED:** tern_stop(...) Internal Function, checks if the most recent coordinate system is ternary, and, if not, stops the current procedure, with a common message format

**DEPRECIATED:** clipPolygons(...) Using the using the PolyClip Package, This clips input polygons for use in the density and contour geometries.

**DEPRECIATED:** theme_arrowbaseline(...) The ternary arrows can have an offset unit value (see tern.axis.arrow.sep), however, it is convenient to set this relative to either the axis, ticks or axis ticklabels (since the latter two can be hidden / removed.). This function permits this to be set

**DEPRECIATED:** element_ternary(...) Replaced by individual theme elements:

1. tern.axis.arrow.show
2. tern.axis.padding
3. tern.axis.arrow.sep
4. tern.axis.arrow.start
5. tern.axis.arrow.finish
6. tern.axis.vshift
7. tern.axis.hshift
8. tern.axis.ticks.length.major
9. tern.axis.ticks.length.minor

**DEPRECIATED:** ggtern.multi is a function which permits the arrangement of multiple ggtern or ggplot2 objects, plots can be provided to the elipsis argument, or, as a list and at the simplest case, the number of columns can be specified. For more advanced usage, consider the layout argument.

**DEPRECIATED:** The point.in.sequence function takes numeric input vectors x and y or a data.frame object, and orders the values in such way that they are correctly sequenced by the angle subtended between each point, and, the centroid of the total set. If the data is provided in the format of a data.frame, then it must containing columns named x and y, else an error will be thrown.

Usage

tern_stop(src = "target")

clipPolygons(df, coord, plyon = c("level", "piece", "group"),
    op = "intersection")
theme_arrowsbaseline(label = "labels")

element_ternary(showarrows, padding, arrowsep, arrowstart, arrowfinish, vshift, hshift, ticklength.major, ticklength.minor)

ggtern.multi(..., plotlist = NULL, cols = 1, layout = NULL)

point.in.sequence(x, y, ..., df = data.frame(x = x, y = y), close = FALSE)

Arguments

src character name of current procedure
df a data frame
coord a ternary coordinate system
plyon items in the data frame to pass to `ddply` argument
op operation method to clip, intersection, union, minus or xor
label a character ('axis', 'ticks' or 'labels') or numeric (rounded to 0, 1 or 2) value to determine the relative location (labels is default) if a character is provided, and it is not one of the above, an error will be thrown.
showarrows logical whether to show the axis directional arrows DEPRECATED
padding the padding around the plot area to make provision for axis labels, ticks and arrows, relative to the cartesian plane. DEPRECATED
arrowsep the distance between ternary axis and ternary arrows DEPRECATED
arrowstart the proportion along the ternary axis to start the directional arrow DEPRECATED
arrowfinish the proportion along the ternary axis to stop the directional arrow DEPRECATED
vshift shift the plot area vertically DEPRECATED
hshift shift the plot area horizontally DEPRECATED
ticklength.major the length of the major ternary ticks as an euclidean distance relative to the x and y limits of the cartesian plot area. DEPRECATED
ticklength.minor the length of the minor ternary ticks as an euclidean distance relative to the x and y limits of the cartesian plot area. DEPRECATED
... additional arguments, multiple plot objects
plotlist alternative to the ... argument, provide a list of `ggplot` or `grob` objects, objects which do not inherit the `ggplot` or `grob` classes will be stripped.
cols number of columns if the layout parameter is not provided.
layout override number of cols, and provide a matrix specifying the layout
x vector of numeric x values
y vector of numeric y values
close logical value (default FALSE), as to whether the set should be closed by adding (duplicating) the first row (after ordering) to the end of the set.
Details

Used to define the layout of some of the ggtern plot features which are unique to the ternary diagrams, and hence, this package.

By default, 1 column is specified, which means that the plots will be stacked on top of each other in a single column, however, if say 4 plots are provided to the ellipsis or plotlist, with cols equal to 2, then this will produce a 2 x 2 arrangement.

In regards to the layout argument (which overrides the cols argument), if it is something like matrix(c(1,2,3,3), nrow=2, byrow=TRUE), then plot number 1 will go in the upper left, 2 will go in the upper right, and 3 will go all the way across the bottom - see the last example below.

The arguments x and y represent cartesian coordinates. This is useful if a path is sought that passes through each point in the ordered set, however, no two lines in the total path cross over each other. Uses the atan2 function to determine the angle (theta) between each point (x,y) and the centroid of the data, it then orders based on increasing values of theta.

Value

data.frame object containing the re-ordered input set.

Author(s)

Nicholas Hamilton

Source

http://www.cookbook-r.com/Graphs/Multiple_graphs_on_one_page_(ggplot2)/
Index

*Topic clipping
  zzz-depreciated, 85

*Topic datasets
  annotation_raster_tern, 6
  coord_tern, 11
  geom_confidence_tern, 18
  geom_crosshair_tern, 20
  geom_density_tern, 22
  geom_errorbarX, 24
  geom_hex_tern, 27
  geom_interpolate_tern, 29
  geom_label_viewport, 31
  geom_mask, 33
  geom_mean_ellipse, 34
  geom_point_swap, 36
  geom_polygon_closed, 37
  geom_smooth_tern, 38
  geom_text_viewport, 40
  geom_tri_tern, 42
  geom_Xisoprop, 44
  geom_Xline, 45
  position_jitter_tern, 60
  position_nudge_tern, 60

*Topic depreciated
  zzz-depreciated, 85

*Topic hplot
  ggplot, 47

*Topic polygon
  zzz-depreciated, 85
  .getFunctions, 4

acomp, 23
aes, 4, 4
aes(), 19, 21, 22, 25, 27, 29, 31, 34, 36–38, 40, 42, 44, 46
aes_(), 19, 21, 22, 25, 27, 29, 31, 34, 36–38, 40, 42, 44, 46
annotate, 5, 5
annotation_raster_tern, 6
approved_geom (approved_layers), 7
approved_layers, 7, 62
approved_position (approved_layers), 7
approved_stat (approved_layers), 7
arrangeGrob, 9
atan2, 87
Atomic or Weight Percent Arrow Label Suffix., 69
atomic_percent
  (ggttern_labels_arrow_suffix), 52
bandwidth.nrd, 19, 23
borders(), 19, 21, 23, 25, 28, 30, 32, 35, 36, 38, 39, 41, 43
breaks_tern, 10, 75
character, 51, 72, 73
clipPolygons (zzz-depreciated), 85
Clockwise/Anticlockwise Axis Precession, 69
Complete Themes, 70
compositions, 34
Confidence Intervals via the Mahalanobis Distance, 54
constructor (ggttern), 49
data.frame, 85
data.frame, 85
custom_percent
  (ggttern_labels_arrow_suffix), 52

Data, 12
Data, 12
Data, 12

88
INDEX

89

data_Feldspar, 12
data_Fragments, 13
data_SkyeLava, 14
data_USDA, 15
data_WhiteCells, 16
draw_key_crosshair_tern (draw_key_tern), 17
draw_key_Liso (draw_key_tern), 17
draw_key_Lline (draw_key_tern), 17
draw_key_Lmark (draw_key_tern), 17
draw_key_point_swap (draw_key_tern), 17
draw_key_Riso (draw_key_tern), 17
geometric_ternary (zzz-depreciated), 85
Elkin and Groves Feldspar Data, 55
Enable/Disable parsing of labels according to latex markup, 69
expression, 51
Feldspar (data_Feldspar), 12
FeldsparRaster (data_Feldspar), 12
fortify(), 19, 21, 22, 25, 27, 29, 32, 34, 36, 37, 39, 40, 43, 44, 46, 47, 50
Fragments (data_Fragments), 13
geom_blank, 8
geom_confidence, 8
geom_confidence (geom_confidence_tern), 18
geom_confidence_tern, 18
gom_count, 7
gom_crosshair_tern, 8, 20
gom_curve, 8
gom_density_tern, 8, 22
gom_errorbar, 24
geom_errorbarh, 24
gom_errorbarL, 7
geom_errorbarL (geom_errorbarX), 24
gom_errorbarR, 7
gom_errorbarR (geom_errorbarX), 24
gom_errorbarT, 7
gom_errorbarT (geom_errorbarX), 24
geom_errorbarX, 24
gom_hex_tern, 8, 27
gom_hline, 45
geom_interpolate_tern, 8, 29
gom_jitter, 7, 8
geom_label, 7, 33
geom_label_viewport, 8, 31
gom_line, 7
gom_Lisoprop, 8
gom_Lisoprop (geom_Xisoprop), 44
gom_Lline, 7
gom_Lline (geom_Xline), 45
gom_Lmark, 8
gom_Lmark (geom_crosshair_tern), 20
geom_mask, 8, 33
gom_mean_ellipses, 8, 34
gom_path, 7
geom_point, 7
geom_point_swap, 8, 36
gom_polygon, 7
gom_polygon_closed, 8, 37
gom_raster, 6
geom_rect, 8
geom_Risoprop, 8
geom_Risoprop (geom_Xisoprop), 44
geom_Rline, 7
geom_Rline (geom_Xline), 45
geom_Rmark, 8
geom_Rmark (geom_crosshair_tern), 20
geom_segment, 7, 11
geom_smooth_tern, 8, 38
geom_text, 7, 42
geom_text_viewport, 8, 40
geom_Tisoprop, 8
geom_Tisoprop (geom_Xisoprop), 44
geom_Tline, 7
geom_Tline (geom_Xline), 45
geom_Tmark, 8
geom_Tmark (geom_crosshair_tern), 20
geom_tri_tern, 8, 42
geom_vline, 45
geom_Xisoprop, 44
geom_Xline, 45
GeomConfidenceTern
(geom_confidence_tern), 18
GeomCrosshairTern
  (geom_crosshair_tern), 20
GeomDensityTern (geom_density_tern), 22
GeomErrorbar1 (geom_errorbarX), 24
GeomErrorbarr (geom_errorbarX), 24
GeomErrorbart (geom_errorbarX), 24
GeomHexTern (geom_hex_tern), 27
GeomInterpolateTern
  (geom_interpolate_tern), 29
GeomLabelViewport
  (geom_label_viewport), 31
GeomLisoprop (geom_Xisoprop), 44
GeomLline (geom_Xline), 45
GeomRmark (geom_crosshair_tern), 20
GeomMeanEllipse (geom_confidence_tern), 18
GeomPointSwap (geom_point_swap), 36
GeomPolygonClosed
  (geom_polygon_closed), 37
GeomRasterAnnTern
  (annotation_raster_tern), 6
GeomRisoprop (geom_Xisoprop), 44
GeomRline (geom_Xline), 45
GeomRmark (geom_crosshair_tern), 20
GeomSmoothTern (geom_smooth_tern), 38
GeomTextViewport
  (geom_text_viewport), 40
GeomTisoprop (geom_Xisoprop), 44
GeomTline (geom_Xline), 45
GeomTmark (geom_crosshair_tern), 20
limit_tern, 84
limit_tern (tern_limits), 64
limits_tern (tern_limits), 64
list, 54
Llab (ggtern_labels), 50
llab (ggtern_labels), 50
Lline (geom_Xline), 45
logical, 51, 73
mahalanobis_distance, 59
margin, 72, 73
mgcv::gam(), 30, 39
multi (zzz-depreciated), 85
multiplot (zzz-depreciated), 85
numeric, 51
percent_atomic
  (ggtern_labels_arrow_suffix), 52
percent_custom
  (ggtern_labels_arrow_suffix), 52
percent_weight
  (ggtern_labels_arrow_suffix), 52
plot.ggplot (ggplot), 47
point.in.sequence (zzz-depreciated), 85
polyclip (zzz-depreciated), 85
position_identity, 9

HERE, 50, 54, 69

label_formatter, 59
labels_tern, 58
labs, 52
Larrowlab (ggtern_labels), 50
larrowlab (ggtern_labels), 50
latex2exp, 51, 74
layer(), 5, 19, 21, 23, 25, 27, 30, 32, 35, 36, 38, 39, 41, 43, 45, 46
limit_tern, 84
limit_tern (tern_limits), 64
limits_tern (tern_limits), 64
list, 54
Llab (ggtern_labels), 50
llab (ggtern_labels), 50
Lline (geom_Xline), 45
logical, 51, 73
mahalanobis_distance, 59
margin, 72, 73
mgcv::gam(), 30, 39
multi (zzz-depreciated), 85
multiplot (zzz-depreciated), 85
numeric, 51
percent_atomic
  (ggtern_labels_arrow_suffix), 52
percent_custom
  (ggtern_labels_arrow_suffix), 52
percent_weight
  (ggtern_labels_arrow_suffix), 52
plot.ggplot (ggplot), 47
point.in.sequence (zzz-depreciated), 85
polyclip (zzz-depreciated), 85
position_identity, 9

HERE, 50, 54, 69

label_formatter, 59
labels_tern, 58
labs, 52
Larrowlab (ggtern_labels), 50
larrowlab (ggtern_labels), 50
latex2exp, 51, 74
layer(), 5, 19, 21, 23, 25, 27, 30, 32, 35, 36, 38, 39, 41, 43, 45, 46
limit_tern, 84
limit_tern (tern_limits), 64
limits_tern (tern_limits), 64
list, 54
Llab (ggtern_labels), 50
llab (ggtern_labels), 50
Lline (geom_Xline), 45
logical, 51, 73
mahalanobis_distance, 59
margin, 72, 73
mgcv::gam(), 30, 39
multi (zzz-depreciated), 85
multiplot (zzz-depreciated), 85
numeric, 51
percent_atomic
  (ggtern_labels_arrow_suffix), 52
percent_custom
  (ggtern_labels_arrow_suffix), 52
percent_weight
  (ggtern_labels_arrow_suffix), 52
plot.ggplot (ggplot), 47
point.in.sequence (zzz-depreciated), 85
polyclip (zzz-depreciated), 85
position_identity, 9

HERE, 50, 54, 69

label_formatter, 59
labels_tern, 58
labs, 52
Larrowlab (ggtern_labels), 50
larrowlab (ggtern_labels), 50
latex2exp, 51, 74
layer(), 5, 19, 21, 23, 25, 27, 30, 32, 35, 36, 38, 39, 41, 43, 45, 46
limit_tern, 84
limit_tern (tern_limits), 64
limits_tern (tern_limits), 64
list, 54
Llab (ggtern_labels), 50
llab (ggtern_labels), 50
Lline (geom_Xline), 45
logical, 51, 73
mahalanobis_distance, 59
margin, 72, 73
mgcv::gam(), 30, 39
multi (zzz-depreciated), 85
multiplot (zzz-depreciated), 85
numeric, 51
percent_atomic
  (ggtern_labels_arrow_suffix), 52
percent_custom
  (ggtern_labels_arrow_suffix), 52
percent_weight
  (ggtern_labels_arrow_suffix), 52
plot.ggplot (ggplot), 47
point.in.sequence (zzz-depreciated), 85
polyclip (zzz-depreciated), 85
position_identity, 9

HERE, 50, 54, 69

label_formatter, 59
labels_tern, 58
labs, 52
Larrowlab (ggtern_labels), 50
larrowlab (ggtern_labels), 50
latex2exp, 51, 74
layer(), 5, 19, 21, 23, 25, 27, 30, 32, 35, 36, 38, 39, 41, 43, 45, 46
limit_tern, 84
limit_tern (tern_limits), 64
limits_tern (tern_limits), 64
list, 54
Llab (ggtern_labels), 50
llab (ggtern_labels), 50
Lline (geom_Xline), 45
logical, 51, 73
mahalanobis_distance, 59
margin, 72, 73
mgcv::gam(), 30, 39
multi (zzz-depreciated), 85
multiplot (zzz-depreciated), 85
numeric, 51
percent_atomic
  (ggtern_labels_arrow_suffix), 52
percent_custom
  (ggtern_labels_arrow_suffix), 52
percent_weight
  (ggtern_labels_arrow_suffix), 52
plot.ggplot (ggplot), 47
point.in.sequence (zzz-depreciated), 85
polyclip (zzz-depreciated), 85
position_identity, 9

INDEX
INDEX

position_jitter_tern, 9, 60, 61
position_nudge_tern, 9, 60, 60
PositionJitterTern
  (position_jitter_tern), 60
PositionNudgeTern
  (position_nudge_tern), 60
predictdf2d, 61
pretty, 75
print.ggplot (ggplot), 47
proto, 54
Rarrowlab (ggtern_labels), 50
rarrowlab (ggtern_labels), 50
Rlab (ggtern_labels), 50
rlab (ggtern_labels), 50
Rline (geom_Xline), 45
rline (geom_Xline), 45
Rotate the plot by X degrees or radians, 69
scale_L_continuous, 65
scale_L_continuous
  (scale_X_continuous), 61
scale_R_continuous, 65
scale_R_continuous
  (scale_X_continuous), 61
scale_T_continuous, 65
scale_T_continuous
  (scale_X_continuous), 61
scale_X_continuous, 61
scales::extended_breaks(), 62
Set Length of arrows, 69
Show/Hide Arrows, 69
Show/Hide Axis Ticklabels, 69
Show/Hide Axis Titles, 69
Show/Hide Grids, 69
Show/Hide Primary/Secondary Ticks, 69
SkyeLava (data_SkyeLava), 14
sprintf, 58
stat_confidence, 8
stat_confidence (geom_confidence_tern), 18
stat_confidence_tern
  (geom_confidence_tern), 18
stat_contour, 20, 35
stat_density_tern, 8
stat_density_tern (geom_density_tern), 22
stat_hex_tern, 8
stat_identity, 8
stat_interpolate_tern, 8
stat_interpolate_tern
  (geom_interpolate_tern), 29
stat_mean_ellipse, 8
stat_mean_ellipse (geom_mean_ellipse), 34
stat_smooth_tern, 8
stat_smooth_tern (geom_smooth_tern), 38
stat_sum, 8
stat_tri_tern, 8
stat_tri_tern (geom_tri_tern), 42
stat_unique, 8
StatConfidenceTern
  (geom_confidence_tern), 18
StatDensityTern (geom_density_tern), 22
StatHexTern (geom_hex_tern), 27
StatInterpolateTern
  (geom_interpolate_tern), 29
StatMeanEllipse (geom_mean_ellipse), 34
stats::loess(), 30, 39
StatSmoothTern (geom_smooth_tern), 38
StatTriTern (geom_tri_tern), 42
strip_unapproved, 62
Tarrowlab (ggtern_labels), 50
tarrowlab (ggtern_labels), 50
tern_anticlockwise (theme_clockwise), 68
tern_clockwise (theme_clockwise), 68
tern_counterclockwise
  (theme_clockwise), 68
tern_limit (tern_limits), 64
tern_limits, 64, 75
tern_stop (zzz-depreciated), 85
Ternary Errorbars, 54
Ternary Theme Elements, 70
ternary_transformation, 63
TeX, 74
theme, 71
theme_anticlockwise (theme_clockwise), 68
theme_arrowbaseline (zzz-depreciated), 85
theme_arrowcustomlength
  (theme_arrowlength), 65
theme_arrowdefault (theme_arrowlength), 65
theme_arrowable (theme_arrowlength), 65
theme_arrowablelarge (theme_arrowlength), 65
theme_showprimary, 80
theme_showsecondary
  (theme_showprimary), 80
theme_showticks (theme_showprimary), 80
theme_showtitles, 82
theme_tern_nogrid (theme_showgrid), 78
theme_tern_nogrid_major
  (theme_showgrid), 78
theme_tern_nogrid_minor
  (theme_showgrid), 78
theme_ticklength, 82
theme_ticklength_major
  (theme_ticklength), 82
theme_ticklength_minor
  (theme_ticklength), 82
theme_ticksinside (theme_ticksoutside), 83
theme_ticksoutside, 83
theme_tropical (ggtern_themes), 55
theme_tropical(...), 69
theme_void (ggtern_themes), 55
theme_zoom (theme_zoom_X), 83
theme_zoom_center (theme_zoom_X), 83
theme_zoom_L (theme_zoom_X), 83
theme_zoom_M (theme_zoom_X), 83
theme_zoom_R (theme_zoom_X), 83
theme_zoom_T (theme_zoom_X), 83
theme_zoom_X, 83
Ticks Inside or Outside of the Main
  Plot Area, 69
Tlab (ggtern_labels), 50
tlab (ggtern_labels), 50
tline (geom_Xline), 45	
line (geom_Xline), 45
tlr2xy, 63
tlr2xy (ternary_transformation), 63
transformation object, 62
Turn off the clipping mask, 69

unit, 71–73
USDA (data_USDA), 15
USDA Textural Classification Data, 55

weight_percent, 51
weight_percent
  (ggtern_labels_arrow_suffix), 52
WhiteCells (data_WhiteCells), 16
Wlab (ggtern_labels), 50