Package ‘ggtern’

December 19, 2018

Version 3.1.0
Date 2018-12-19
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 (>= 3.1.0)
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
Enhances sp
License GPL-2 | file LICENSE
Encoding UTF-8
URL http://www.ggtern.com

Collate 'ggtern-package.R' 'aes.R' 'coord-tern.R' 'calc-tern-tlr2xy.R'
'calc-mahalanobis-distance.R' 'calc-kde2d-weighted.R'
'doc-data.R' 'doc-theme-convenience.R' 'depreciated.R'
'labels-new.R' 'labels-percent.R' 'legend-draw-tern.R'
'ggtern-constructor.R' 'gg-internal.R'
'modifications-gridExtra.R' 'onLoad.R' 'plot.R' 'plot-build.R'
'plot-construction.R' 'position-tern.R' 'position-nudge-tern.R'
'position-jitter-tern.R' 'save.R' 'scales-tern.R'
'strip-unapproved.R' 'tern-limits.R' 'theme.R'
'theme-arrowlength.R' 'theme-clockwise.R' 'theme-defaults.R'
'theme-elements.R' 'theme-gridsontop.R' 'theme-bordersontop.R'
'theme-legend-position.R' 'theme-latex.R' 'theme-mesh.R'
'theme-noarrows.R' 'theme-nomask.R' 'theme-novar-tern.R'
'theme-rotate.R' 'theme-showgrid.R' 'theme-showlabels.R'
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-viewpost.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 2018-12-19 11:20:03 UTC

R topics documented:

+.gg .................................................. 4
.getFunctions ........................................ 4
aes .................................................. 5
annotate ............................................. 5
annotation_raster_tern .............................. 6
approved_layers ..................................... 7
arrangeGrob ......................................... 9
breaks_tern ......................................... 11
coord_tern ......................................... 11
data_Feldspar ....................................... 12
data_Fragments ...................................... 13
data_SkyeLava ...................................... 15
data_USDA .......................................... 16
data_WhiteCells .................................... 17
draw_key_tern ....................................... 18
geom_confidence_tern ................................ 19
geom_crosshair_tern .................................. 21
geom_density_tern ................................... 23
geom_errorbarX ..................................... 25
geom_hex_tern ...................................... 28
geom_interpolate_tern .............................. 30
geom_label_viewport ............................... 32
geom_mask .......................................... 34
geom_mean_ellipse .................................. 35
geom_point_swap .................................... 37
geom_polygon_closed ............................... 38
### R topics documented:

- `geom_smooth_tern` .......................... 39
- `geom_text_viewport` ...................... 41
- `geom_tri_tern` ............................. 43
- `geom_Xisoprop` ......................... 45
- `geom_Xline` ............................... 46
- `ggplot` .................................. 48
- `ggsave` .................................. 49
- `ggtern` .................................. 50
- `ggtern_labels` ............................ 51
- `ggtern_labels_arrow_suffix` ............ 53
- `ggtern_package` ......................... 54
- `ggtern_themes` ............................ 56
- `labels_tern` .............................. 59
- `label_formatter` .......................... 60
- `mahalanobis_distance` .................. 60
- `position_jitter_tern` ................... 61
- `position_nudge_tern` .................... 61
- `predictdf2d` ............................. 62
- `scale_X_continuous` ..................... 62
- `strip_unapproved` ....................... 63
- `ternary_transformation` .................. 64
- `tern_limits` ............................. 65
- `theme` .................................. 66
- `theme_arrowlength` ...................... 75
- `theme_bordersontop` ..................... 77
- `theme_clockwise` ......................... 78
- `theme_complete` ........................... 78
- `theme_convenience_functions` ......... 79
- `theme_elements` ........................... 81
- `theme_gridsontop` ....................... 83
- `theme_latex` ............................. 83
- `theme_legend_position` ................. 84
- `theme_mesh` .............................. 85
- `theme_noarrows` ......................... 85
- `theme_nomask` ............................ 86
- `theme_novar_tern` ....................... 86
- `theme_rotate` ............................ 87
- `theme_showgrid` .......................... 88
- `theme_showlabels` ....................... 89
- `theme_showprimary` ...................... 90
- `theme_showtitles` ....................... 91
- `theme_ticklength` ....................... 91
- `theme_ticksoutside` ..................... 92
- `theme_zoom_X` ............................ 92
- `zzz-depreciated` ......................... 94

### Index

- 97
Description

"+.gg" is a local copy of the ggplot2 add function, no change other than exporting from the ggtern namespace.

Usage

```r
## S3 method for class 'gg'
e1 + e2
e1 %>% e2
```

Arguments

- `e1`: first object
- `e2`: second object

Author(s)

Nicholas Hamilton

Description

OLD FUNCTIONS new_panel', 'train_layout', 'train_position', 'train_ranges', 'map_position', 'map',
`xlabel`, 'ylabel'

Usage

`.getFunctions()`
**Modified Aesthetic Mappings**

**Description**

Modified Aesthetic Mappings

**Usage**

```r
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

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_rbgw() +
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 ggtern are 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
- `ncol`: argument of gtable
- `widths`: argument of gtable
- `heights`: argument of gtable
- `top`: optional string, or grob
- `bottom`: optional string, or grob
- `left`: optional string, or grob
- `right`: optional string, or grob
- `padding`: unit of length one, margin around annotations
- `newpage`: open a new page

Author(s)

Nicholas Hamilton
**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)
```

---

**coord_tern**  
*Ternary Coordinate System*

**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**

```r
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, \textit{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
data_Fragments

References


See Also

Data

Examples

# Summarize the Feldspar Data
data(feldspar)
summary(feldspar)

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

---

Granham 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 watershed. 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

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=.75,alpha=.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)
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

```r
# 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)
b 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_2O_3",
      Rarrow = "M = MgO")
```
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

**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)
**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**

```r
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
Description

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

Usage

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)

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 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.

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).
geom_confidence_tern

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().

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). If NULL, estimated using bandwidth.nrd.

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

Aesthetics

gem_ConfidenceTernunderstands the following aesthetics (required aesthetics are in bold):

• x
• y
• alpha
• 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()
**geom_crosshair_tern**  

**Ternary Crosshairs**

**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**

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

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

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

```r
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.

- **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

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

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.

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.

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

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

linemitre Line mitre limit (number greater than 1).
geom_density_tern

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().

gem 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.

bd1 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]

bd1.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 bd1 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

# 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()

Description

ggeom_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

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

- **null**: 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.

- **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_errorbarX)

- **geom_errorbarX** understands the following aesthetics (required aesthetics are in bold):
  
  - **Tmax**
  - **Tmin**
  - **x**
  - **y**
  - **z**
  - **alpha**
  - **colour**
  - **linetype**
  - **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

# 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')
**geom_hex_tern**

*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)
```

```r
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.

- **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.

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

A 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

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
ggtern(df, aes(x,y,z)) + geom_interpolate_tern(binwidth=0.05, fun=myfun)
```

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.
- `stat` Use to override the default connection between `geom_smooth()` and `stat_smooth()`.
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 a character vector, e.g. "auto", "lm", "glm", "gam", "loess" or a function, e.g. `MASS::rlm` or `mgcv::gam`, `base::lm`, or `base::loess`.

For `method = "auto"` the smoothing method is chosen based on the size of the largest group (across all panels). `loess()` is used for less than 1,000 observations; otherwise `mgcv::gam()` is used with formula `y ~ s(x, bs = "cs")`. 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 = "auto"` 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)`

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

- x
- y
- alpha
- colour
- linetype
- size
Author(s)
Nicholas Hamilton

Examples

```r
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()
```

**geom_label_viewport**  
*Draw Label 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 label-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

```r
geom_label_viewport(mapping = NULL, data = NULL, stat = "identity",
  position = "identity", ..., hjust = "inward", vjust = "inward",
  parse = FALSE, label.padding = unit(0.25, "lines"),
  label.r = unit(0.15, "lines"), label.size = 0.25, 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.
- **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**

Author(s)

Nicholas Hamilton
geom_mask

See Also

geom_label

Examples

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=1.0,y=0.0,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

geom_mask()

Author(s)

Nicholas Hamilton
**Examples**

```r
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**

```r
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.
- **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_mean_ellipse 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

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() +
  stat_mean_ellipse(geom='polygon',steps=500,fill='red',color='black') +
  geom_point()
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**

```r
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.

- `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')

geom_polygon_closed     Closed Polygons

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.

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 adjust-
          ment function.
geom_smooth_tern

... 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

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.
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.

Position adjustment, either as a string, or the result of a call to a position adjust-
ment 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.

Smoothing method (function) to use, accepts either a character vector, e.g. "auto",
"lm", "glm", "gam", "loess" or a function, e.g. MASS::rlm or mgcv::gam,
base::lm, or base::loess.
For method = "auto" the smoothing method is chosen based on the size of the
largest group (across all panels). loess() is used for less than 1,000 observa-
tions; otherwise mgcv::gam() is used with formula = y ~ s(x, bs = "cs").
Somewhat anecdotally, loess gives a better appearance, but is O(N^2) in mem-
ory, 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 = "auto" would use, then set method = "gam", formula = y ~ s(x, bs = "cs").
Formula to use in smoothing function, eg. y ~ x, y ~ poly(x, 2), y ~ log(x)
Display confidence interval around smooth? (TRUE by default, see level to
control.)
If FALSE, the default, missing values are removed with a warning. If TRUE,
missing values are silently removed.
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.
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 the range of values by this much (vector of length 2) when fullrange is
set to TRUE
Number of points at which to evaluate smoother.
Controls the amount of smoothing for the default loess smoother. Smaller num-
bers produce wigglier lines, larger numbers produce smoother lines.
Should the fit span the full range of the plot, or just the data?
Level of confidence interval to use (0.95 by default).
List of additional arguments passed on to the modelling function defined by
method.
**geom_text_viewport**

**Author(s)**

Nicholas Hamilton

**Examples**

```r
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**

```r
geom_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.
- **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.
- **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

### See Also

`geom_text`
**Examples**

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

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

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

---

**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)
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.
- **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()`. 
  - If `mapping` is specified, then the data is inherited from `mapping`.
  - If neither is specified, then 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.

**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)) +
  geom_tri_tern(bins=5,aes(fill=..stat..,value=wt),fun=mean) +
  geom_point(size=0.25)
```

**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.

... 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 isopropotion 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**

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`
geom_Xline

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)
lline(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()`. 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.

- **...**
  - 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_fline(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.
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.
limitsize  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)
```

```

ggtern  ggtern Constructor

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)
```
- ```r
llab(label, labelarrow = label)
```
- ```r
rlab(label, labelarrow = label)
```
- ```r
wlab(label)
```
- ```r
zlab(label)
```
- ```r	arrowlab(label)
```
Larrowlab(label)
Rarrowlab(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**

AAA lab takes precedence over BBB lab (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**

Aliasses exist for Tlab, Llab, Rlab and Wlab, which are tlab, llab, rlab and wlab. These aliasses 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 \texttt{latex2exp} package, and these formats can be parsed too.
**ggtern_labels_arrow_suffix**

**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_30_8",
     xarrow = "Albite, NaAlSi_30_8",
     y = "(Na,K)AlSi_30_8",
     yarrow = "Anorthite (Na,K)AlSi_30_8",
     z = "KAlSi_30_8",
     zarrow = "Orthoclase KAlSi_30_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 Wilkinson’s “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 Wickham’s 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 hierarchies 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 modified 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.
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
labels_tern

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 = "%.2f", 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
labels_tern()
labels_tern(limits = c(0,.5))

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

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

Usage
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
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**

```
xL yL zL  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**

```
xL yL zL  Amount of compositions to nudge
```
**scale_X_continuous**

**Author(s)**

Nicholas Hamilton

**See Also**

Other position adjustments: position_jitter_tern

---

**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**

```r
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

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

limits {A numeric vector of length two providing limits of the scale. Use \texttt{NA} to refer to the existing minimum or maximum.}

breaks {One of:
  - \texttt{NULL} for no breaks
  - \texttt{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

minor_breaks {One of:
  - \texttt{NULL} for no minor breaks
  - \texttt{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:
  - \texttt{NULL} for no labels
  - \texttt{waiver()} for the default labels computed by the transformation object
  - A character vector giving labels (must be same length as \texttt{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

\texttt{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 \texttt{approved_layers} for the current list of approved geometries and stats

Usage

\texttt{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

Author(s)

Nicholas Hamilton
**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**

```r
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

```r
# 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..)) +
guides(color = 'red', 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

### Modify components of a theme (ggtern version)

#### Description

Use `theme()` to modify individual components of a theme, allowing you to control the appearance of all non-data components of the plot. `theme()` only affects a single plot: see `theme_update()` if you want modify the active theme, to affect all subsequent plots.

#### Usage

```r
theme(line, rect, text, title, aspect.ratio, axis.title, axis.title.x,
      axis.title.x.top, axis.title.y, axis.title.y.right, axis.text,
      axis.text.x, axis.text.x.top, axis.text.y, axis.text.y.right, axis.ticks,
      axis.ticks.x, axis.ticks.y, axis.ticks.length, axis.line, axis.line.x,
      axis.line.y, legend.background, legend.margin, legend-spacing,
      legend-spacing.x, legend-spacing.y, legend.key, legend.key.size,
      legend.key.height, legend.key.width, legend-text, legend-text-align,
      legend.title, legend.title.align, legend-position, legend-direction,
      legend-justification, legend-box, legend-box.just, legend-box.margin,
      legend-box-background, legend-box-spacing, panel-background,
      panel.border, panel-spacing, panel-spacing.x, panel-spacing.y,
      panel-grid, panel-grid-major, panel-grid.minor, panel-grid-major.x,
      panel-grid-major.y, panel-grid.minor.x, panel-grid.minor.y, panel-ontop,
      plot-background, plot-title, plot-subtitle, plot-caption, plot-margin,
```
Arguments

line all line elements (element_line())
rect all rectangular elements (element_rect())
text all text elements (element_text())
title all title elements: plot, axes, legends (element_text(); inherits from text)
aspect.ratio aspect ratio of the panel
axis.title labels of axes (element_text()). Specify all axes’ labels (axis.title), labels by plane (using axis.title.x or axis.title.y), or individually for each axis (using axis.title.x.bottom, axis.title.x.top, axis.title.y.left, axis.title.y.right). axis.title.*.* inherits from axis.title.* which inherits from axis.title, which in turn inherits from text
axis.title.x labels of axes (element_text()). Specify all axes’ labels (axis.title), labels by plane (using axis.title.x or axis.title.y), or individually for each axis (using axis.title.x.bottom, axis.title.x.top, axis.title.y.left, axis.title.y.right). axis.title.*.* inherits from axis.title.* which inherits from axis.title, which in turn inherits from text
axis.title.x.top labels of axes (element_text()). Specify all axes’ labels (axis.title), labels by plane (using axis.title.x or axis.title.y), or individually for each axis (using axis.title.x.bottom, axis.title.x.top, axis.title.y.left,
theme

axis.title.y.right) axis.title.* inherits from axis.title.*, which inherits from text

axis.title.y labels of axes (element_text()). Specify all axes’ labels (axis.title), labels by plane (using axis.title.x or axis.title.y), or individually for each axis (using axis.title.x.bottom, axis.title.x.top, axis.title.y.left, axis.title.y.right). axis.title.* inherits from axis.title.*, which inherits from axis.title, which in turn inherits from text

axis.title.y.right labels of axes (element_text()). Specify all axes’ labels (axis.title), labels by plane (using axis.title.x or axis.title.y), or individually for each axis (using axis.title.x.bottom, axis.title.x.top, axis.title.y.left, axis.title.y.right). axis.title.* inherits from axis.title.*, which inherits from axis.title, which in turn inherits from text

axis.text tick labels along axes (element_text()). Specify all axis tick labels (axis.text), tick labels by plane (using axis.text.x or axis.text.y), or individually for each axis (using axis.text.x.bottom, axis.text.x.top, axis.text.y.left, axis.text.y.right). axis.text.* inherits from axis.text.*, which inherits from axis.text, which in turn inherits from text

axis.text.x tick labels along axes (element_text()). Specify all axis tick labels (axis.text), tick labels by plane (using axis.text.x or axis.text.y), or individually for each axis (using axis.text.x.bottom, axis.text.x.top, axis.text.y.left, axis.text.y.right). axis.text.* inherits from axis.text.*, which inherits from axis.text, which in turn inherits from text

axis.text.x.top tick labels along axes (element_text()). Specify all axis tick labels (axis.text), tick labels by plane (using axis.text.x or axis.text.y), or individually for each axis (using axis.text.x.bottom, axis.text.x.top, axis.text.y.left, axis.text.y.right). axis.text.* inherits from axis.text.*, which inherits from axis.text, which in turn inherits from text

axis.text.y tick labels along axes (element_text()). Specify all axis tick labels (axis.text), tick labels by plane (using axis.text.x or axis.text.y), or individually for each axis (using axis.text.x.bottom, axis.text.x.top, axis.text.y.left, axis.text.y.right). axis.text.* inherits from axis.text.*, which inherits from axis.text, which in turn inherits from text

axis.text.y.right tick labels along axes (element_text()). Specify all axis tick labels (axis.text), tick labels by plane (using axis.text.x or axis.text.y), or individually for each axis (using axis.text.x.bottom, axis.text.x.top, axis.text.y.left, axis.text.y.right). axis.text.* inherits from axis.text.*, which inherits from axis.text, which in turn inherits from text

axis.ticks tick marks along axes (element_line()). Specify all tick marks (axis.ticks), ticks by plane (using axis.ticks.x or axis.ticks.y), or individually for each axis (using axis.ticks.x.bottom, axis.ticks.x.top, axis.ticks.y.left, axis.ticks.y.right). axis.ticks.* inherits from axis.ticks.*, which inherits from axis.ticks, which in turn inherits from line

axis.ticks.x tick marks along axes (element_line()). Specify all tick marks (axis.ticks), ticks by plane (using axis.ticks.x or axis.ticks.y), or individually for each axis (using axis.ticks.x.bottom, axis.ticks.x.top, axis.ticks.y.left, axis.ticks.y.right). axis.ticks.* inherits from axis.ticks.*, which inherits from axis.ticks, which in turn inherits from line
axis (using `axis.ticks.x.bottom, axis.ticks.x.top, axis.ticks.y.left, axis.ticks.y.right`). `axis.ticks.*.*` inherits from `axis.ticks.*` which inherits from `axis.ticks`, which in turn inherits from `line`

axis.ticks.y  
tick marks along axes (**element_line()**). Specify all tick marks (axis.ticks), ticks by plane (using `axis.ticks.x` or `axis.ticks.y`), or individually for each axis (using `axis.ticks.x.bottom, axis.ticks.x.top, axis.ticks.y.left, axis.ticks.y.right`). `axis.ticks.*.*` inherits from `axis.ticks.*` which inherits from `axis.ticks`, which in turn inherits from `line`

axis.ticks.length  
length of tick marks (unit)

axis.line  
lines along axes (**element_line()**). Specify lines along all axes (axis.line), lines for each plane (using `axis.line.x` or `axis.line.y`), or individually for each axis (using `axis.line.x.bottom, axis.line.x.top, axis.line.y.left, axis.line.y.right`). `axis.line.*.*` inherits from `axis.line.*` which inherits from `axis.line`, which in turn inherits from `line`

axis.line.x  
lines along axes (**element_line()**). Specify lines along all axes (axis.line), lines for each plane (using `axis.line.x` or `axis.line.y`), or individually for each axis (using `axis.line.x.bottom, axis.line.x.top, axis.line.y.left, axis.line.y.right`). `axis.line.*.*` inherits from `axis.line.*` which inherits from `axis.line`, which in turn inherits from `line`

axis.line.y  
lines along axes (**element_line()**). Specify lines along all axes (axis.line), lines for each plane (using `axis.line.x` or `axis.line.y`), or individually for each axis (using `axis.line.x.bottom, axis.line.x.top, axis.line.y.left, axis.line.y.right`). `axis.line.*.*` inherits from `axis.line.*` which inherits from `axis.line`, which in turn inherits from `line`

legend.background  
background of legend (**element_rect()**; inherits from `rect`)

legend.margin  
the margin around each legend (**margin()**)

legend.spacing  
the spacing between legends (unit). `legend.spacing.x` & `legend.spacing.y` inherit from `legend.spacing` or can be specified separately

legend.spacing.x  
the spacing between legends (unit). `legend.spacing.x` & `legend.spacing.y` inherit from `legend.spacing` or can be specified separately

legend.spacing.y  
the spacing between legends (unit). `legend.spacing.x` & `legend.spacing.y` inherit from `legend.spacing` or can be specified separately

legend.key  
background underneath legend keys (**element_rect()**; inherits from `rect`)

legend.key.size  
size of legend keys (unit); key background height & width inherit from `legend.key.size` or can be specified separately

legend.key.height  
size of legend keys (unit); key background height & width inherit from `legend.key.size` or can be specified separately

legend.key.width  
size of legend keys (unit); key background height & width inherit from `legend.key.size` or can be specified separately
theme

- **legend.text** legend item labels (`element_text();` inherits from `text`)
- **legend.text.align** alignment of legend labels (number from 0 (left) to 1 (right))
- **legend.title** title of legend (`element_text();` inherits from `title`)
- **legend.title.align** alignment of legend title (number from 0 (left) to 1 (right))
- **legend.position** the position of legends ("none", "left", "right", "bottom", "top", or two-element numeric vector)
- **legend.direction** layout of items in legends ("horizontal" or "vertical")
- **legend.justification** anchor point for positioning legend inside plot ("center" or two-element numeric vector) or the justification according to the plot area when positioned outside the plot
- **legend.box** arrangement of multiple legends ("horizontal" or "vertical")
- **legend.box.just** justification of each legend within the overall bounding box, when there are multiple legends ("top", "bottom", "left", or "right")
- **legend.box.margin** margins around the full legend area, as specified using `margin()`
- **legend.box.background** background of legend area (`element_rect();` inherits from `rect`)
- **legend.box.spacing** The spacing between the plotting area and the legend box (unit)
- **panel.background** background of plotting area, drawn underneath plot (`element_rect();` inherits from `rect`)
- **panel.border** border around plotting area, drawn on top of plot so that it covers tick marks and grid lines. This should be used with `fill = NA` (`element_rect();` inherits from `rect`)
- **panel.spacing** spacing between facet panels (unit). `panel.spacing.x` & `panel.spacing.y` inherit from `panel.spacing` or can be specified separately.
- **panel.spacing.x** spacing between facet panels (unit). `panel.spacing.x` & `panel.spacing.y` inherit from `panel.spacing` or can be specified separately.
- **panel.spacing.y** spacing between facet panels (unit). `panel.spacing.x` & `panel.spacing.y` inherit from `panel.spacing` or can be specified separately.
- **panel.grid** grid lines (`element_line()`). Specify major grid lines, or minor grid lines separately (using `panel.grid.major` or `panel.grid.minor`) or individually for each axis (using `panel.grid.major.x`, `panel.grid.minor.x`, `panel.grid.major.y`, `panel.grid.minor.y`). Y axis grid lines are horizontal and x axis grid lines are vertical. `panel.grid.*.*` inherits from `panel.grid.*` which inherits from `panel.grid`, which in turn inherits from `line`
panel.grid.major

grid lines (element_line()). Specify major grid lines, or minor grid lines separately (using panel.grid.major or panel.grid.minor) or individually for each axis (using panel.grid.major.x, panel.grid.minor.x, panel.grid.major.y, panel.grid.minor.y). Y axis grid lines are horizontal and x axis grid lines are vertical. panel.grid.* inherits from panel.grid.* which inherits from panel.grid, which in turn inherits from line

panel.grid.minor

grid lines (element_line()). Specify major grid lines, or minor grid lines separately (using panel.grid.major or panel.grid.minor) or individually for each axis (using panel.grid.major.x, panel.grid.minor.x, panel.grid.major.y, panel.grid.minor.y). Y axis grid lines are horizontal and x axis grid lines are vertical. panel.grid.* inherits from panel.grid.* which inherits from panel.grid, which in turn inherits from line

panel.grid.major.x

grid lines (element_line()). Specify major grid lines, or minor grid lines separately (using panel.grid.major or panel.grid.minor) or individually for each axis (using panel.grid.major.x, panel.grid.minor.x, panel.grid.major.y, panel.grid.minor.y). Y axis grid lines are horizontal and x axis grid lines are vertical. panel.grid.* inherits from panel.grid.* which inherits from panel.grid, which in turn inherits from line

panel.grid.major.y

grid lines (element_line()). Specify major grid lines, or minor grid lines separately (using panel.grid.major or panel.grid.minor) or individually for each axis (using panel.grid.major.x, panel.grid.minor.x, panel.grid.major.y, panel.grid.minor.y). Y axis grid lines are horizontal and x axis grid lines are vertical. panel.grid.* inherits from panel.grid.* which inherits from panel.grid, which in turn inherits from line

panel.grid.minor.x

grid lines (element_line()). Specify major grid lines, or minor grid lines separately (using panel.grid.major or panel.grid.minor) or individually for each axis (using panel.grid.major.x, panel.grid.minor.x, panel.grid.major.y, panel.grid.minor.y). Y axis grid lines are horizontal and x axis grid lines are vertical. panel.grid.* inherits from panel.grid.* which inherits from panel.grid, which in turn inherits from line

panel.grid.minor.y

grid lines (element_line()). Specify major grid lines, or minor grid lines separately (using panel.grid.major or panel.grid.minor) or individually for each axis (using panel.grid.major.x, panel.grid.minor.x, panel.grid.major.y, panel.grid.minor.y). Y axis grid lines are horizontal and x axis grid lines are vertical. panel.grid.* inherits from panel.grid.* which inherits from panel.grid, which in turn inherits from line

panel.on top

option to place the panel (background, gridlines) over the data layers (logical). Usually used with a transparent or blank panel.background.

plot.background

background of the entire plot (element_rect(); inherits from rect)

plot.title

plot title (text appearance) (element_text(); inherits from title) left-aligned by default
plot.subtitle plot subtitle (text appearance) \texttt{(element_text()); inherits from title) left-aligned by default
plot.caption caption below the plot (text appearance) \texttt{(element_text()); inherits from title) right-aligned by default
plot.margin margin around entire plot (unit with the sizes of the top, right, bottom, and left margins)

strip.background background of facet labels \texttt{(element_rect()); inherits from rect). Horizontal facet background \texttt{(strip.background.x)} & vertical facet background \texttt{(strip.background.y)} inherit from \texttt{strip.background} or can be specified separately

strip.placemenet placement of strip with respect to axes, either "inside" or "outside". Only important when axes and strips are on the same side of the plot.

strip.text facet labels \texttt{(element_text()); inherits from text). Horizontal facet labels \texttt{(strip.text.x)} & vertical facet labels \texttt{(strip.text.y)} inherit from \texttt{strip.text} or can be specified separately

strip.text.x facet labels \texttt{(element_text()); inherits from text). Horizontal facet labels \texttt{(strip.text.x)} & vertical facet labels \texttt{(strip.text.y)} inherit from \texttt{strip.text} or can be specified separately

strip.text.y facet labels \texttt{(element_text()); inherits from text). Horizontal facet labels \texttt{(strip.text.x)} & vertical facet labels \texttt{(strip.text.y)} inherit from \texttt{strip.text} or can be specified separately

strip.switch.pad.grid space between strips and axes when strips are switched (unit)

strip.switch.pad.wrap space between strips and axes when strips are switched (unit)

tern.axis.arrow Base Arrow Line \texttt{('element_line'; inherits from 'axis.line')}

tern.axis.arrow.T Arrow Line for TOP Axis \texttt{('element_line'; inherits from 'tern.axis.arrow')}

tern.axis.arrow.L Arrow Line for LHS Axis \texttt{('element_line'; inherits from 'tern.axis.arrow')}

tern.axis.arrow.R Arrow Line for RHS Axis \texttt{('element_line'; inherits from 'tern.axis.arrow')}

tern.axis.arrow.text Base Arrow Label \texttt{('element_text'; inherits from 'tern.axis.text')}

tern.axis.arrow.text.T Arrow Label on TOP Axis \texttt{('element_text'; inherits from 'tern.axis.arrow.text')}

tern.axis.arrow.text.L Arrow Label on LHS Axis \texttt{('element_text'; inherits from 'tern.axis.arrow.text')}

tern.axis.arrow.text.R Arrow Label on RHS Axis \texttt{('element_text'; inherits from 'tern.axis.arrow.text')}

tern.axis.arrow.start Proportion of Axis when Arrow Starts \texttt{('numeric')}
tern.axis.arrow.finish
Proportion of Axis when Arrow Finishes ('numeric')

tern.axis.arrow.sep
Arrows Separation from Axis ('numeric')

tern.axis.arrow.show
Arrows Show or Hide ('logical')

tern.axis.clockwise
Clockwise or Anticlockwise Precession ('logical')

tern.axis.vshift
Amount to nudge the plot vertically ('numeric')

tern.axis.hshift
Amount to nudge the plot horizontally ('numeric')

tern.axis.line.ontop
Bring Axis Borders on Top of Everything (Deprecated) ('logical')

tern.axis.line
Base Line ('element_line'; inherits from 'axis.line')

tern.axis.line.T
Line for TOP Axis ('element_line'; inherits from 'tern.axis.line')

tern.axis.line.L
Line for LHS Axis ('element_line'; inherits from 'tern.axis.line')

tern.axis.line.R
Line for RHS Axis ('element_line'; inherits from 'tern.axis.line')

tern.axis.text
Base Text ('element_text'; inherits from 'axis.text')

tern.axis.text.T
Text for TOP Axis ('element_text'; inherits from 'tern.axis.text')

tern.axis.text.L
Text for LHS Axis ('element_text'; inherits from 'tern.axis.text')

tern.axis.text.R
Text for RHS Axis ('element_text'; inherits from 'tern.axis.text')

tern.axis.text.show
Axis Labels Show or Hide ('logical')

tern.axis.ticks
Base Ticks ('element_line'; inherits from 'axis.ticks')

tern.axis.ticks.length.major
Ticks Major Ticklength ('unit')

tern.axis.ticks.length.minor
Ticks Minor Ticklength ('unit')

tern.axis.ticks.major
Base Major Ticks ('element_line'; inherits from 'tern.axis.ticks')

tern.axis.ticks.major.T
Base Major Ticks for TOP Axis ('element_line'; inherits from 'tern.axis.ticks.major')

tern.axis.ticks.major.L
Base Major Ticks for LHS Axis ('element_line'; inherits from 'tern.axis.ticks.major')

tern.axis.ticks.major.R
Base Major Ticks for RHS Axis ('element_line'; inherits from 'tern.axis.ticks.major')
tern.axis.ticks.minor
    Base Minor Ticks ('element_line'; inherits from 'tern.axis.ticks')
tern.axis.ticks.minor.T
    Base Minor Ticks for TOP Axis ('element_line'; inherits from 'tern.axis.ticks.minor')
tern.axis.ticks.minor.L
    Base Minor Ticks for LHS Axis ('element_line'; inherits from 'tern.axis.ticks.minor')
tern.axis.ticks.minor.R
    Base Minor Ticks for RHS Axis ('element_line'; inherits from 'tern.axis.ticks.minor')
tern.axis.ticks.outside
    Ticks Outside or Inside ('logical')
tern.axis.ticks.primary.show
    Ticks Show Primary ('logical')
tern.axis.ticks.secondary.show
    Ticks Show Secondary ('logical')
tern.axis.title
    Base Apex Title ('element_text'; inherits from 'axis.title')
tern.axis.title.T
    Apex Title for TOP Axis ('element_text'; inherits from 'tern.axis.title')
tern.axis.title.L
    Apex Title for LHS Axis ('element_text'; inherits from 'tern.axis.title')
tern.axis.title.R
    Apex Title for RHS Axis ('element_text'; inherits from 'tern.axis.title')
tern.axis.title.show
    Apex Titles Show or Hide ('logical')
tern.panel.background
    Background of Ternary Plot Area** ('element_rect'; inherits from 'panel.background')
tern.panel.expand
    The amount to expand the ternary plotting panel, in ratio to npc units ('numeric')
tern.panel.grid.major
    Base Major Gridline ('element_line'; inherits from 'panel.grid.major')
tern.panel.grid.major.T
    Major Gridline for TOP Axis ('element_line'; inherits from 'tern.panel.grid.major')
tern.panel.grid.major.L
    Major Gridline for LHS Axis ('element_line'; inherits from 'tern.panel.grid.major')
tern.panel.grid.major.R
    Major Gridline for RHS Axis ('element_line'; inherits from 'tern.panel.grid.major')
tern.panel.grid.major.show
    Show or Hide Major Gridline ('logical')
tern.panel.grid.minor
    Base Minor Gridline ('element_line'; inherits from 'panel.grid.minor')
tern.panel.grid.minor.T
    Minor Gridline for TOP Axis ('element_line'; inherits from 'tern.panel.grid.minor')
tern.panel.grid.minor.L
    Minor Gridline for LHS Axis ('element_line'; inherits from 'tern.panel.grid.minor')
theme_arrowlength

tern.panel.grid.minor.R
  Minor Gridline for RHS Axis (‘element_line’, inherits from ‘tern.panel.grid.minor’)
tern.panel.grid.minor.show
  Show or Hide Minor Gridline (‘logical’)
tern.panel.grid.ontop
  Bring grids, axis and axis labels on top of everything else (‘logical’)
tern.panel.mask.show
  Show or Hide the Clipping Mask (‘logical’)
tern.panel.rotate
  The amount to rotate the ternary diagram in degrees (‘numeric’)
tern.plot.background
  Background of Ternary Clipping Area** (‘element_rect’, inherits from ‘plot.background’)
tern.plot.latex
  Whether to parse characters as latex commands (‘logical’)

... additional element specifications not part of base ggplot2. If supplied validate needs to be set to FALSE.

complete
  set this to TRUE if this is a complete theme, such as the one returned by theme_grey(). Complete themes behave differently when added to a ggplot object. Also, when setting complete = TRUE all elements will be set to inherit from blank elements.

validate
  TRUE to run validate_element(), FALSE to bypass checks.

Theme inheritance

Theme elements 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’. This means that you can modify the appearance of multiple elements by setting a single high-level component.

Author(s)

Nicholas Hamilton

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 aliases), 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_arrowshort()

theme_arrownormal()

theme_arrowdefault()

theme_arrownormal()

theme_arrownormal()

Arguments

start        a numeric scalar, or numeric vector of length three (3), representing the fractional [0,1] position along the axis where the arrow/s should START.

finish       a numeric scalar, or numeric vector of length three (3), representing the fractional [0,1] position along the axis where the arrow/s should FINISH.

Details

If the ternary arrows are switched OFF (via the theme_hidearrows command, or the theme(tern.axis.arrow.show=FALSE) theme element), then under such circumstance, these convenience functions will turn ON the ternary arrows, essentially running theme_showarrows or theme(tern.axis.arrow.show=TRUE)

If for some reason, the start and finish arguments are identical, then the ternary arrows will be switched OFF, tantamount to running the theme_hidearrows convenience function.

Custom Length

theme_arrowcustomlength or theme_arrowlength (alias) sets the ternary arrow lengths to values as specified by the user, occupying a length between the values as specified by the start and finish arguments (fractions) relative to the length of the ternary axis.

Short Arrow Length

theme_arrowsmall or theme_arrowshort(alias) 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 (alias) 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 (alias) increases the ternary arrows to long arrows occupying a length between 0.2 and 0.8 of the length of the ternary axis.

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_arrownormal()
plot + theme_arrowsmall()
plot + theme_arrowcustomlength(.1,.8)
plot + theme_arrowlarge()
plot + theme_arrowcustomlength(.1,.8)
plot + theme_arrowsmalllength(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_clockwise  \textit{Direction of Ternary Rotation}

**Description**

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

**Usage**

\begin{verbatim}
theme_clockwise()

theme_anticlockwise()

theme_counterclockwise()
\end{verbatim}

**Details**

If the \texttt{tern.axis.arrow.show} value is \texttt{FALSE}, these functions will set it to \texttt{TRUE}.

**Author(s)**

Nicholas Hamilton

\hline

\begin{center}
theme_complete  \textit{List of Available Themes}
\end{center}

**Description**

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

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

**Author(s)**
Nicholas Hamilton

**See Also**

`ggtern_themes`

---

## Theme Convenience Functions

**Description**

`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 [HERE](#).

**Convenience Functions**

Some of the Convenience functions that ship with `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>tern.panel.background</td>
<td><code>element_rect/(panel.background)</code></td>
<td>Background of Ternary Plot Area**</td>
</tr>
<tr>
<td>tern.plot.background</td>
<td><code>element_rect/(plot.background)</code></td>
<td>Background of Ternary Clipping Area**</td>
</tr>
<tr>
<td>tern.plot.latex</td>
<td><code>logical</code></td>
<td>Whether to parse characters as latex commands</td>
</tr>
<tr>
<td>tern.axis.hshift</td>
<td><code>numeric</code></td>
<td>Amount to nudge the plot horizontally</td>
</tr>
<tr>
<td>tern.axis.vshift</td>
<td><code>numeric</code></td>
<td>Amount to nudge the plot vertically</td>
</tr>
<tr>
<td>tern.axis.clockwise</td>
<td><code>logical</code></td>
<td>Clockwise or Anticlockwise Precession</td>
</tr>
<tr>
<td>tern.axis.line</td>
<td><code>element_line/(axis.line)</code></td>
<td>Base Line</td>
</tr>
<tr>
<td>tern.axis.line.T</td>
<td><code>element_line/(tern.axis.line)</code></td>
<td>Line for TOP Axis</td>
</tr>
<tr>
<td>tern.axis.line.L</td>
<td><code>element_line/(tern.axis.line)</code></td>
<td>Line for LHS Axis</td>
</tr>
<tr>
<td>tern.axis.line.R</td>
<td><code>element_line/(tern.axis.line)</code></td>
<td>Line for RHS Axis</td>
</tr>
<tr>
<td>tern.axis.line.on top</td>
<td><code>logical</code></td>
<td>Bring Axis Borders on Top of Everything (Depreciated)</td>
</tr>
<tr>
<td>tern.axis.text</td>
<td><code>element_text/(axis.text)</code></td>
<td>Base Text</td>
</tr>
<tr>
<td>tern.axis.text.T</td>
<td><code>element_text/(tern.axis.text)</code></td>
<td>Text for TOP Axis</td>
</tr>
<tr>
<td>tern.axis.text.L</td>
<td><code>element_text/(tern.axis.text)</code></td>
<td>Text for LHS Axis</td>
</tr>
<tr>
<td>tern.axis.text.R</td>
<td><code>element_text/(tern.axis.text)</code></td>
<td>Text for RHS Axis</td>
</tr>
<tr>
<td>tern.axis.text.show</td>
<td><code>logical</code></td>
<td>Axis Labels Show or Hide</td>
</tr>
<tr>
<td>tern.axis.title</td>
<td><code>element_text/(axis.title)</code></td>
<td>Base Apex Title</td>
</tr>
<tr>
<td>tern.axis.title.T</td>
<td><code>element_text/(tern.axis.title)</code></td>
<td>Apex Title for TOP Axis</td>
</tr>
<tr>
<td>tern.axis.title.L</td>
<td><code>element_text/(tern.axis.title)</code></td>
<td>Apex Title for LHS Axis</td>
</tr>
<tr>
<td>tern.axis.title.R</td>
<td><code>element_text/(tern.axis.title)</code></td>
<td>Apex Title for RHS Axis</td>
</tr>
<tr>
<td>tern.axis.title.show</td>
<td><code>logical</code></td>
<td>Apex Titles Show or Hide</td>
</tr>
<tr>
<td>tern.axis.arrow</td>
<td><code>element_line/(axis.line)</code></td>
<td>Base Arrow Line</td>
</tr>
<tr>
<td>tern.axis.arrow.T</td>
<td><code>element_line/(tern.axis.arrow)</code></td>
<td>Arrow Line for TOP Axis</td>
</tr>
<tr>
<td>tern.axis.arrow.L</td>
<td><code>element_line/(tern.axis.arrow)</code></td>
<td>Arrow Line for LHS Axis</td>
</tr>
<tr>
<td>tern.axis.arrow.R</td>
<td><code>element_line/(tern.axis.arrow)</code></td>
<td>Arrow Line for RHS Axis</td>
</tr>
</tbody>
</table>
theme_elements

** 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
**theme_gridsontop**

**Render Grids on Top**

**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()
```

**Author(s)**

Nicholas Hamilton

---

**theme_latex**

**Parse Labels w Latex Markup**

**Description**

A series of convenience functions that either enable or disable the use of the 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**

```
theme_latex(value = TRUE)
theme_showlatex()
theme_nolatex()
theme_hidelatex()
```

**Arguments**

```
value logical as to whether to enable latex parsing or not
```

**Author(s)**

Nicholas Hamilton
See Also

TeX

Examples

```r
# 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)

# Demonstrate latex overriding the bold face
ggtern() +
  labs(title = '\textit{Plot Title}') +
  theme_latex(TRUE) + 
  theme('plot.title' = element_text(face='bold'))

# Turn off latex parsing, bold title revealed
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

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()
### 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.

**Usage**

```r
theme_nomask()
theme_hidemask()
theme_showmask()
```

### 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**

```r
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

    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, ie 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 \textit{disables} MAJOR gridlines.
theme_showgrid_major is a function which \textit{enables} MINOR gridlines.
theme_hidegrid_minor or theme_nogrid_minor (alias) is a function which \textit{disables} MINOR gridlines.
theme_showgrid is a function which \textit{enables} both MAJOR and MINOR gridlines.

\textbf{Author(s)}

Nicholas Hamilton

\textbf{Examples}

```r
#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()
```

\textbf{theme_showlabels} \hspace{1cm} \textit{Show or Hide Axis Ticklabels}

\textbf{Description}

Convenience functions to enable or disable the axis ticklabels

\textbf{Usage}

```r
theme_showlabels()
```

\textbf{Details}

theme_showlabels is a function that appends to the current theme a flag to switch ON the axis ticklabels, whilst theme_hidelabels or theme_nolabels (Alias) are functions that appends to the current theme a flag to switch OFF the axis ticklabels

\textbf{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.
theme_showtitles

**Author(s)**
Nicholas Hamilton

**Examples**
```
data(feldspar)
plot <- ggtern(data=Feldspar,aes(An,Ab,Or)) + geom_point() + 
  theme_showsecondary()
```

---

**theme_showtitles**  
*Show or Hide the Axis (Apex) Titles*

**Description**
Convenience functions to SHOW or HIDE the apex labels.

**Usage**
- `theme_showtitles`
- `theme_hidetitles`
- `theme_notitles`

**Author(s)**
Nicholas Hamilton

**Examples**
```
#Load data
data(Feldspar)
ggtern(data=Feldspar,aes(An,Ab,Or)) + geom_point() + theme_bw() + theme_hidetitles()
```

---

**theme_ticklength**  
*Modify the Ticklengths*

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

**Usage**
- `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.

Author(s)

Nicholas Hamilton

Examples

```r
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

```r
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 93

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() +
        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")
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 ellipsis 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

```r
tern_stop(src = "target")

clipPolygons(df, coord, plyon = c("level", "piece", "group"),
              op = "intersection")```
theme-arrowbaseline(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 DEPRECI-
ATED
arrowfinish the proportion along the ternary axis to stop the directional arrow DEPRECI-
ATED
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, 94

*Topic datasets
  annotation_raster_tern, 6
  coord_tern, 11
  geom_confidence_tern, 19
  geom_crosshair_tern, 21
  geom_density_tern, 23
  geom_errorbarX, 25
  geom_hex_tern, 28
  geom_interpolate_tern, 30
  geom_label_viewport, 32
  geom_mask, 34
  geom_mean_ellipse, 35
  geom_point_swap, 37
  geom_polygon_closed, 38
  geom_smooth_tern, 39
  geom_text_viewport, 41
  geom_tri_tern, 43
  geom_xisoprop, 45
  geom_xline, 46
  position_jitter_tern, 61
  position_nudge_tern, 61

*Topic depreciated
  zzz-depreciated, 94

*Topic hplot
  ggplot, 48

*Topic polygon
  zzz-depreciated, 94

+.gg, 4
.getFunctions, 4
%+% (+.gg), 4

acomp, 24
aes, 5, 5
aes(, 19, 21, 23, 25, 28, 30, 32, 35, 37–39, 41, 43, 45, 47
aes_(), 19, 21, 23, 25, 28, 30, 32, 35, 37–39, 41, 43, 45, 47
annotate, 5, 6

annotation_raster_tern, 6
approved_geom(approved_layers), 7
approved_layers, 7, 63
approved_position(approved_layers), 7
approved_stat(approved_layers), 7
arrangeGrob, 9
atan2, 96
atomic_percent
  (ggtern_labels_arrow_suffix), 53

bandwidth.nrd, 20, 24
borders(), 20, 22, 24, 26, 29, 31, 33, 36, 37, 39, 40, 42, 44
breaks_tern, 11, 85

class, 52
clipPolygons(zzz-depreciated), 94
compositions, 35
creator (ggtern), 50
custom_percent
  (ggtern_labels_arrow_suffix), 53

data, 13
data.frame, 94
data_Feldspar, 12
data_Fragments, 13
data_SkyleLava, 15
data_USDA, 16
data_WhiteCells, 17
draw_key_crosshair_tern
  (draw_key_tern), 18
draw_key_Liso(draw_key_tern), 18

Data, 13
data.frame, 94
data_Feldspar, 12
data_Fragments, 13
data_SkyleLava, 15
data_USDA, 16
data_WhiteCells, 17
draw_key_crosshair_tern
  (draw_key_tern), 18
draw_key_Liso(draw_key_tern), 18
draw_key_Lline (draw_key_tern), 18
draw_key_Lmark (draw_key_tern), 18
draw_key_point_swap (draw_key_tern), 18
draw_key_Riso (draw_key_tern), 18
draw_key_Rline (draw_key_tern), 18
draw_key_Rmark (draw_key_tern), 18
draw_key_Tern (draw_key_tern), 18
draw_key_Tiso (draw_key_tern), 18
draw_key_Tline (draw_key_tern), 18
draw_key_Tmark (draw_key_tern), 18
drawmahal, 60

element_blank, 88
element_line, 81, 82, 88
element_line(), 67–71
element_rect, 81, 82
element_rect(), 67, 69–72
element_ternary (zzz-deprecated), 94
element_text, 81, 82
element_text(), 67, 68, 70–72
expression, 52

Feldspar (data_Feldspar), 12
FeldsparRaster (data_Feldspar), 12
fortify(), 19, 21, 23, 26, 28, 30, 32, 35, 37, 38, 40, 41, 44, 45, 47, 48, 51
Fragments (data_Fragments), 13
gemm blank, 8
gemm confidence, 8
gemm confidence (geom confidence_tern), 19
gemm confidence_tern, 19
gemm count, 8
gemm crosshair_tern, 8, 21
gemm curve, 8
gemm density_tern, 8, 23
gemm errorbar, 25
gemm errorbarh, 25
gemm errorbarL, 8
gemm errorbarL (geom errorbarX), 25
gemm errorbarR, 8
gemm errorbarR (geom errorbarX), 25
gemm errorbarT, 8
gemm errorbarT (geom errorbarX), 25
gemm errorbarX, 25
gemm_hex_tern, 8, 28
gemhline, 46
gem_interpolate_tern, 8, 30

gemm jitter, 8
gemm label, 8, 34
gemm_label_tern, 8, 32
gemm_line, 8
gemm_Lisoprop, 8
gemm_Lisoprop (geom_Xisoprop), 45
gemm_Lline, 8
gemm_Lisoprop (geom_Lisoprop), 45
gemm_Lline (geom_Lline), 46
gemm_Lmark, 8
gemm_Lmark (geom_crosshair_tern), 21
gemm_mask, 8, 34
gemm_mean_ellipse, 8, 35
gemm_path, 8
gemm_point, 8
gemm_point_swap, 8, 37
gemm_polygon, 8
gemm_polygon_closed, 8, 38
gemm_raster, 6
gemm_rect, 8
gemm_Risoprop, 8
gemm_Risoprop (geom_Xisoprop), 45
gemm_Rline, 8
gemm_Rline (geom_Xline), 46
gemm_Rmark, 8
gemm_Rmark (geom_crosshair_tern), 21
gemm_segment, 8, 12
gemm_smooth_tern, 8, 39
gemm_text, 8, 42
gemm_text_viewport, 8, 41
gemm_Tisoprop, 8
gemm_Tisoprop (geom_Xisoprop), 45
gemm_Tline, 8
gemm_Tline (geom_Xline), 46
gemm_Tmark, 8
gemm_Tmark (geom_crosshair_tern), 21
gemm_tri_tern, 8, 43
gemm_vline, 46
gemm_Xisoprop, 45
gemm_Xline, 46
GeomConfidenceTern
 (geom_confidence_tern), 19
GeomCrosshairTern
 (geom_crosshair_tern), 21
GeomDensityTern (geom_density_tern), 23
GeomErrorbarL (geom_errorbarX), 25
GeomErrorbarR (geom_errorbarX), 25
GeomErrorbarT (geom_errorbarX), 25
GeomErrorbarX, 25
GeomHexTern (geom_hex_tern), 28
INDEX

GeomInterpolateTern
  (geom_interpolate_tern), 30
GeomLabelViewport
  (geom_label_viewport), 32
GeomLisoprop (geom_Xisoprop), 45
GeomLine (geom_Xline), 46
GeomLmark (geom_crosshair_tern), 21
GeomMask (geom_mask), 34
GeomMeanEllipse (geom_confidence_tern), 19
GeomPointSwap (geom_point_swap), 37
GeomPolygonClosed
  (geom_polygon_closed), 38
GeomRasterAnnTern
  (annotation_raster_tern), 6
GeomRisoprop (geom_Xisoprop), 45
GeomRLine (geom_Xline), 46
GeomRmark (geom_crosshair_tern), 21
GeomSmoothTern (geom_smooth_tern), 39
GeomTextViewport (geom_text_viewport), 41
GeomTisoprop (geom_Xisoprop), 45
GeomTLine (geom_Xline), 46
GeomTmark (geom_crosshair_tern), 21
GeomTriTern (geom_hex_tern), 28
getBreaks (breaks_tern), 11
getLabels (labels_tern), 59
ggplot, 48, 51
ggplot(). 19, 21, 23, 26, 28, 30, 32, 35, 37, 38, 40, 41, 43, 45, 47
ggplot2, 7, 46, 54, 55
ggplot_build, 49
ggsave, 49
ggtern, 50
ggtern datasets, 16
ggtern-labels (ggtern_labels), 51
ggtern-package (ggtern_package), 54
ggtern_multi (zzz-depreciated), 94
ggtern_labels, 51
ggtern_labels_arrow_suffix, 53
ggtern_package, 54
ggtern_themes, 56, 79
grid, 55
grid.arrange (arrangeGrob), 9
grid.draw.ggplot (ggsave), 49
HERE, 51, 55, 79
label_formatter, 60
labels_tern, 59
labs, 53
Larrowlab (ggtern_labels), 51
larrowlab (ggtern_labels), 51
latex2exp, 52, 83
layer(), 6, 19, 21, 23, 26, 28, 31, 33, 36, 37, 39, 40, 42, 44, 46, 47
limit_tern, 93
limit_tern (tern_limits), 65
limits_tern (tern_limits), 65
list, 55
Llab (ggtern_labels), 51
llab (ggtern_labels), 51
lline (geom_Xline), 46
lline (geom_Xline), 46
loess(), 31, 40
logical, 52, 81, 82
mahalanobis_distance, 60
margin(), 69, 70
mgcv::gam(), 31, 40
multi (zzz-depreciated), 94
multiplot (zzz-depreciated), 94
numeric, 52, 81, 82
percent_atomic
  (ggtern_labels_arrow_suffix), 53
percent_custom
  (ggtern_labels_arrow_suffix), 53
percent_weight
  (ggtern_labels_arrow_suffix), 53
plot.ggplot (ggplot), 48
point.in.sequence (zzz-depreciated), 94
polyclip (zzz-depreciated), 94
position_identity, 9
position_jitter_tern, 9, 61, 62
position_nudge_tern, 9, 61, 61
PositionJitterTern
  (position_jitter_tern), 61
PositionNudgeTern
  (position_nudge_tern), 61
predictdf2d, 62
pretty, 85
print.ggplot (ggplot), 48
proto, 55
 INDEX

Rarrowlab (ggtern_labels), 51
rarrowlab (ggtern_labels), 51
Rlab (ggtern_labels), 51
rlab (ggtern_labels), 51
Rline (geom_Xline), 46
rline (geom_Xline), 46
scale_L_continuous, 65
scale_L_continuous
(scale_X_continuous), 62
scale_R_continuous, 65
scale_R_continuous
(scale_X_continuous), 62
scale_T_continuous, 65
scale_T_continuous
(scale_X_continuous), 62
scale_X_continuous, 62
SkEyelava (data_SkEyelava), 15
sprintf, 59
stat_confidence, 9
stat_confidence (geom_confidence_tern), 19
stat_confidence_tern
(geom_confidence_tern), 19
stat_contour, 20, 36
stat_density_tern, 9
stat_density_tern (geom_density_tern), 23
stat_hex_tern, 9
stat_hex_tern (geom_hex_tern), 28
stat_identity, 9
stat_interpolate_tern, 9
stat_interpolate_tern
(geom_interpolate_tern), 30
stat_mean_ellipse, 9
stat_mean_ellipse (geom_mean_ellipse), 35
stat_smooth_tern, 9
stat_smooth_tern (geom_smooth_tern), 39
stat_sum, 9
stat_tri_tern, 9
stat_tri_tern (geom_tri_tern), 43
stat_unique, 9
StatConfidenceTern
(geom_confidence_tern), 19
StatDensityTern (geom_density_tern), 23
StatHexTern (geom_hex_tern), 28
StatInterpolateTern
(geom_interpolate_tern), 30
StatMeanEllipse (geom_mean_ellipse), 35
StatSmoothTern (geom_smooth_tern), 39
StatTriTern (geom_tri_tern), 43
strip_unapproved, 63
Tarrowlab (ggtern_labels), 51
tarrowlab (ggtern_labels), 51
tern_anticlockwise (theme_clockwise), 78
tern_clockwise (theme_clockwise), 78
tern_counterclockwise
(theme_clockwise), 78
tern_limit (tern_limits), 65
tern_limits, 65, 85
tern_stop (zzz-deprecated), 94
ternary_transformation, 64
TeX, 83, 84
theme, 66, 81
theme_anticlockwise (theme_clockwise), 78
theme_arrowbaseline (zzz-deprecated), 94
theme_arrowcustomlength
(theme_arrowlength), 75
theme_arroldefault (theme_arrowlength), 75
theme_arroldlarge (theme_arrowlength), 75
theme_arroldlength, 75
theme_arroldlong (theme_arrowlength), 75
theme_arroldnormal (theme_arrowlength), 75
theme_arroldshort (theme_arrowlength), 75
theme_arroldsmall (theme_arrowlength), 75
theme_bluedark (ggtern_themes), 56
theme_bluedark(...), 79
theme_bluelight (ggtern_themes), 56
theme_bluelight(...), 79
theme_bordersonbottom
(theme_bordersontop), 77
theme_bordersonbottom, 77
theme_bvbg (ggtern_themes), 56
theme_bvbg(...), 79
theme_bvbw (ggtern_themes), 56
theme_bvbw(...), 79
theme_bw (ggtern_themes), 56
theme_bw(...), 78
theme_classic (ggtern_themes), 56
theme_classic(...), 78
theme_clockwise, 78
theme_complete, 78
theme_zoom_center (theme_zoom_X), 92
theme_zoom_L (theme_zoom_X), 92
theme_zoom_M (theme_zoom_X), 92
theme_zoom_R (theme_zoom_X), 92
theme_zoom_T (theme_zoom_X), 92
theme_zoom_X, 92
Tlab (ggtern_labels), 51
tlab (ggtern_labels), 51
tline (geom_Xline), 46
tline (geom_Xline), 46
tlr2xy, 64
tlr2xy (ternary_transformation), 64
unit, 82
USDA (data_USDA), 16

weight_percent, 52
weight_percent
   (ggtern_labels_arrow_suffix), 53
WhiteCells (data_WhiteCells), 17
Wlab (ggtern_labels), 51
wlab (ggtern_labels), 51

xlab, 51, 52
xy2tlr, 64
xy2tlr (ternary_transformation), 64

ylab, 51, 52

zlab (ggtern_labels), 51
zzz-deprecated, 94