Package 'gggda'

July 22, 2025

Title A 'ggplot2' Extension for Geometric Data Analysis

Version 0.1.1

Description A variety of multivariable data summary statistics and constructions have been proposed, either to generalize univariable analogs or to exploit multivariable properties. Notable among these are the bivariate peelings surveyed by Green (1981, ISBN:978-0-471-28039-2), the bag-and-bolster plots proposed by Rousseeuw &al (1999) <doi:10.1080/00031305.1999.10474494>, and the minimum spanning trees used by Jolliffe (2002) <doi:10.1007/b98835> to represent high-dimensional relationships among data in a low-dimensional plot. Additionally, biplots of singular value--decomposed tabular data, such as from principal components analysis, make use of vectors, calibrated axes, and other representations of variable elements to complement point markers for case elements; see Gabriel (1971) <doi:10.1093/biomet/58.3.453> and Gower & Harding (1988) <doi:10.1093/biomet/75.3.445> for original proposals. Because they treat the abscissa and ordinate as commensurate or the data elements themselves as point masses or unit vectors, these multivariable tools can be thought of as belonging to geometric data analysis; see Podani (2000, ISBN:90-5782-067-6) for techniques and applications and Le Roux & Rouanet (2005) <doi:10.1007/1-4020-2236-0> for foundations. 'gggda' extends Wickham's (2010) <doi:10.1198/jcgs.2009.07098> layered grammar of graphics with statistical transformation (``stat'') and geometric construction (``geom'') layers for many of these tools, as well as convenience coordinate systems to emphasize intrinsic geometry of the data.

Depends R (>= 3.3.0), ggplot2

Imports rlang, tidyr, dplyr, magrittr, scales, labeling, ddalpha

Suggests gridExtra, MASS, Hmisc, tibble, mlpack, testthat, knitr, rmarkdown

License GPL-3

Encoding UTF-8

URL https://github.com/corybrunson/gggda, https://corybrunson.github.io/gggda/

BugReports https://github.com/corybrunson/gggda/issues

RoxygenNote 7.3.2

Collate 'aaa-.r' 'aes-coord.r' 'coord-rect.r' 'depth-median.r'
'geom-axis.r' 'geom-bagplot.r' 'key-intervals.r'
'geom-intervals.r' 'geom-isoline.r' 'geom-rule.r'
'geom-text-radiate.r' 'geom-utils.r' 'geom-vector.r' 'gggda.r'
'peel.r' 'stat-peel.r' 'stat-depth.r' 'stat-bagplot.r'
'stat-center.r' 'stat-cone.r' 'stat-referent.r' 'stat-rule.r'
'stat-scale.r' 'stat-spantree.r' 'utils.r'

VignetteBuilder knitr

NeedsCompilation no

Author Jason Cory Brunson [aut, cre] (ORCID:
<https://orcid.org/0000-0003-3126-9494>),
Emily Paul [ctb],
John Gracey [aut]

Maintainer Jason Cory Brunson <cornelioid@gmail.com>

Repository CRAN

Date/Publication 2025-07-19 23:50:02 UTC

Contents

aes-coord *Multidimensional coordinate mappings*

Description

Allow stat layers to receive a sequence of positional variables rather than only x and y.

Usage

```
aes_coord(.data, prefix)

get_aes_coord(data)

aes_c(...)
```

Arguments

.data, data	A data frame. .data stands in for the data passed to ggplot2::ggplot(), while data is expected to have been pre-processed before being passed to a Stat*$compute_*() function.
prefix	A regular expression used to identify the coordinate columns of .data.
...	objects to be concatenated. All NULL entries are dropped before method dispatch unless at the very beginning of the argument list.

Details

These functions coordinate (pun intended) the use of more than two positional variables in plot layers. Pass multidimensional coordinates to a stat via mapping = aes_coord(...) and reconcile the recovered coordinates with x and y (which are overridden if present) in Stat*$compute_*(); see the StatChull source code for an example. Use aes_c() to concatenate aesthetic mappings.

Value

A list with class uneval. Components of the list are either quosures or constants.

See Also

ggplot2::aes() for standard **ggplot2** aesthetic mappings.

coord_rect *Cartesian coordinates and plotting window with fixed aspect ratios*

Description

Geometric data analysis often requires that coordinates lie on the same scale. The coordinate system `CoordRect`, alias `CoordSquare`, provides control of both coordinate and window aspect ratios.

Usage

```
coord_rect(
  ratio = 1,
  window_ratio = ratio,
  xlim = NULL,
  ylim = NULL,
  expand = TRUE,
  clip = "on"
)
```

Arguments

`ratio`	aspect ratio, expressed as `y / x`
`window_ratio`	aspect ratio of plotting window
`xlim, ylim`	Limits for the x and y axes.
`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`.
`clip`	Should drawing be clipped to the extent of the plot panel? A setting of `"on"` (the default) means yes, and a setting of `"off"` means no. In most cases, the default of `"on"` should not be changed, as setting `clip = "off"` can cause unexpected results. It allows drawing of data points anywhere on the plot, including in the plot margins. If limits are set via `xlim` and `ylim` and some data points fall outside those limits, then those data points may show up in places such as the axes, the legend, the plot title, or the plot margins.

Value

A `Coord` ggproto object.

Examples

```
# ensures that the resolutions of the axes and the dimensions of the plotting
# window respect the specified aspect ratios
p <- ggplot(mtcars, aes(mpg, hp/10)) + geom_point()
p + coord_rect(ratio = 1)
p + coord_rect(ratio = 1, window_ratio = 2)
p + coord_rect(ratio = 1, window_ratio = 1/2)
```

```
p + coord_rect(ratio = 5)
p + coord_rect(ratio = 1/5)
p + coord_rect(xlim = c(15, 30))
p + coord_rect(ylim = c(15, 30))

# square (even excluding some geometric constructions)
p + coord_square(xlim = c(0, 30), ylim = c(20, 40))
```

`depth_median` *Depth median*

Description

Compute the depth median of a data set.

Usage

```
depth_median(x, notion = "zonoid", ...)
```

Arguments

`x`	Matrix of data whose depth median is to be calculated; see `ddalpha::depth.()`.
`notion`	The name of the depth notion (shall also work with a user-defined depth function named `"depth.<name>"`).
`...`	Additional parameters passed to the depth functions.

Details

This function is called internally by `stat_bagplot()` and can be passed to `stat_center()` but is also exported directly for data analysis.

Value

A one-row matrix of depth median coordinates.

Examples

```
# sample median
iris %>%
  subset(select = -Species) %>%
  depth_median()
# groupwise medians
iris %>%
  split(~ Species) %>%
  lapply(subset, select = -Species) %>%
  lapply(depth_median) %>%
  simplify2array() %>% t() %>% as.data.frame()
```

draw-key *Key drawing functions for bivariate intervals.*

Description

These key drawing functions supplement those built into **ggplot2** for legend glyphs suitable to bivariate line-ranges and point-ranges.

Usage

```
draw_key_line(data, params, size)

draw_key_crosslines(data, params, size)

draw_key_crosspoint(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.

Details

`draw_key_line()` is a horizontal counterpart to `ggplot2::draw_key_vline()`. `draw_key_crosslines()` superimposes these two keys, and `draw_key_crosspoint()` additionally superimposes an oversized `ggplot2::draw_key_point()`.

Value

A grid grob.

See Also

ggplot2::draw_key for key glyphs installed with **ggplot2**.

geom_axis *Axes through or offset from the origin*

Description

geom_axis() renders lines through or orthogonally translated from the origin and the position of each case or variable.

Usage

```
geom_axis(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  axis_labels = TRUE,
  axis_ticks = TRUE,
  axis_text = TRUE,
  by = NULL,
  num = NULL,
  tick_length = 0.025,
  text_dodge = 0.03,
  label_dodge = 0.03,
  ...,
  axis.colour = NULL,
  axis.color = NULL,
  axis.alpha = NULL,
  label.angle = 0,
  label.colour = NULL,
  label.color = NULL,
  label.alpha = NULL,
  tick.linewidth = 0.25,
  tick.colour = NULL,
  tick.color = NULL,
  tick.alpha = NULL,
  text.size = 2.6,
  text.angle = 0,
  text.hjust = 0.5,
  text.vjust = 0.5,
  text.family = NULL,
  text.fontface = NULL,
  text.colour = NULL,
  text.color = NULL,
  text.alpha = NULL,
  parse = FALSE,
  check_overlap = FALSE,
  na.rm = FALSE,
```

```
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by aes(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
stat	The statistical transformation to use on the data for this layer. When using a geom_*() function to construct a layer, the stat argument can be used the override the default coupling between geoms and stats. The stat argument accepts the following: • A Stat ggproto subclass, for example StatCount. • A string naming the stat. To give the stat as a string, strip the function name of the stat_ prefix. For example, to use stat_count(), give the stat as "count". • For more information and other ways to specify the stat, see the layer stat documentation.
position	A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following: • The result of calling a position function, such as position_jitter(). This method allows for passing extra arguments to the position. • A string naming the position adjustment. To give the position as a string, strip the function name of the position_ prefix. For example, to use position_jitter(), give the position as "jitter". • For more information and other ways to specify the position, see the layer position documentation.
axis_labels, axis_ticks, axis_text	Logical; whether to include labels, tick marks, and text value marks along the axes.
by, num	Intervals between elements or number of elements; specify only one.
tick_length	Numeric; the length of the tick marks, as a proportion of the minimum of the plot width and height.
text_dodge	Numeric; the orthogonal distance of tick mark text from the axis, as a proportion of the minimum of the plot width and height.

label_dodge
: Numeric; the orthogonal distance of the axis label from the axis, as a proportion of the minimum of the plot width and height.

...
: Additional arguments passed to ggplot2::layer().

axis.colour, axis.color, axis.alpha
: Default aesthetics for axes. Set to NULL to inherit from the data's aesthetics.

label.angle, label.colour, label.color, label.alpha
: Default aesthetics for labels. Set to NULL to inherit from the data's aesthetics.

tick.linewidth, tick.colour, tick.color, tick.alpha
: Default aesthetics for tick marks. Set to NULL to inherit from the data's aesthetics.

text.size, text.angle, text.hjust, text.vjust, text.family, text.fontface, text.colour, text.color, text.alpha
: Default aesthetics for tick mark labels. Set to NULL to inherit from the data's aesthetics.

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

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

na.rm
: Passed to ggplot2::layer().

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().

Details

Axes are lines that track the values of linear variables across a plot. Multivariate scatterplots may include more axes than plotting dimensions, in which case the plot may display only a fraction of the total variation in the data.

Gower & Hand (1996) recommend using axes to represent numerical variables in biplots. Consequently, Gardner & le Roux (2002) refer to these as Gower biplots.

Axes positioned orthogonally at the origin are a ubiquitous feature of scatterplots and used both to recover variable values from case markers (prediction) and to position new case markers from variables (interpolation). When they are not orthogonal, these two uses conflict, so interpolative versus predictive axes must be used appropriately.

Value

A ggproto layer.

Aesthetics

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

- `x`
- `y`
- `lower`
- `upper`
- `yintercept` *or* `xintercept` *or* `xend` and `yend`
- `linetype`
- `linewidth`
- `size`
- `hjust`
- `vjust`
- `colour`
- `alpha`
- `label`
- `family`
- `fontface`
- `center`, `scale`
- `group`

References

Gower JC & Hand DJ (1996) *Biplots*. Chapman & Hall, ISBN: 0-412-71630-5.

Gardner S, le Roux N (2002) "Biplot Methodology for Discriminant Analysis Based upon Robust Methods and Principal Curves". *Classification, Clustering, and Data Analysis: Recent Advances and Applications*: 169–176. https://link.springer.com/chapter/10.1007/978-3-642-56181-8_18

See Also

Other geom layers: `geom_bagplot()`, `geom_isoline()`, `geom_lineranges()`, `geom_rule()`, `geom_text_radiate()`, `geom_vector()`

Examples

```
# stack loss gradient
stackloss %>%
  lm(formula = stack.loss ~ Air.Flow + Water.Temp + Acid.Conc.) %>%
  coef() %>%
  as.list() %>% as.data.frame() %>%
  subset(select = c(Air.Flow, Water.Temp, Acid.Conc.)) ->
  coef_data
# gradient axis with respect to two predictors
```

```
scale(stackloss, scale = FALSE) %>%
  ggplot(aes(x = Acid.Conc., y = Air.Flow)) +
  coord_square() +
  geom_point(aes(size = stack.loss, alpha = sign(stack.loss))) +
  scale_size_area() + scale_alpha_binned(breaks = c(-1, 0, 1)) +
  geom_axis(data = coef_data)
# unlimited axes with window forcing
stackloss_centered <- scale(stackloss, scale = FALSE)
stackloss_centered %>%
  ggplot(aes(x = Acid.Conc., y = Air.Flow)) +
  coord_square() +
  geom_point(aes(size = stack.loss, alpha = sign(stack.loss))) +
  scale_size_area() + scale_alpha_binned(breaks = c(-1, 0, 1)) +
  stat_rule(
    geom = "axis", data = coef_data,
    referent = stackloss_centered,
    fun.lower = function(x) minpp(x, p = 1),
    fun.upper = function(x) maxpp(x, p = 1),
    fun.offset = function(x) minabspp(x, p = 1)
  )
# NB: `geom_axis(stat = "rule")` would fail to pass positional aesthetics.

# eigen-decomposition of covariance matrix
ability.cov$cov %>%
  cov2cor() %>%
  eigen() %>% getElement("vectors") %>%
  as.data.frame() %>%
  transform(test = rownames(ability.cov$cov)) ->
  ability_cor_eigen
# test axes in best-approximation space
ability_cor_eigen %>%
  transform(E3 = ifelse(V3 > 0, "rise", "fall")) %>%
  ggplot(aes(V1, V2, color = E3)) +
  coord_square() +
  geom_axis(aes(label = test), text.color = "black", text.alpha = .5) +
  expand_limits(x = c(-1, 1), y = c(-1, 1))
```

---

geom_bagplot *Bagplots*

Description

Render bagplots from tagged data comprising medians, hulls, contours, and outlier specifications.

Usage

```
geom_bagplot(
  mapping = NULL,
  data = NULL,
  stat = "bagplot",
```

```
  position = "identity",
  ...,
  bag.linewidth = sync(),
  bag.linetype = sync(),
  bag.colour = "black",
  bag.color = NULL,
  bag.fill = sync(),
  bag.alpha = NA,
  median.shape = 21L,
  median.stroke = sync(),
  median.size = 5,
  median.colour = sync(),
  median.color = NULL,
  median.fill = "white",
  median.alpha = NA,
  fence.linewidth = 0.25,
  fence.linetype = 0L,
  fence.colour = sync(),
  fence.color = NULL,
  fence.fill = sync(),
  fence.alpha = 0.25,
  outlier.shape = sync(),
  outlier.stroke = sync(),
  outlier.size = sync(),
  outlier.colour = sync(),
  outlier.color = NULL,
  outlier.fill = NA,
  outlier.alpha = NA,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

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

`stat` The statistical transformation to use on the data for this layer. When using a `geom_*()` function to construct a layer, the `stat` argument can be used the override the default coupling between geoms and stats. The `stat` argument accepts the following:

- A `Stat` ggproto subclass, for example `StatCount`.
- A string naming the stat. To give the stat as a string, strip the function name of the `stat_` prefix. For example, to use `stat_count()`, give the stat as `"count"`.
- For more information and other ways to specify the stat, see the layer stat documentation.

`position` A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The `position` argument accepts the following:

- The result of calling a position function, such as `position_jitter()`. This method allows for passing extra arguments to the position.
- A string naming the position adjustment. To give the position as a string, strip the function name of the `position_` prefix. For example, to use `position_jitter()`, give the position as `"jitter"`.
- For more information and other ways to specify the position, see the layer position documentation.

`...` Additional arguments passed to `ggplot2::layer()`.

`bag.linetype`, `bag.linewidth`, `bag.colour`, `bag.color`, `bag.fill`, `bag.alpha`
Default aesthetics for bags. Set to `sync()` to inherit from the data's aesthetics or to `NULL` to use the data's aesthetics.

`median.shape`, `median.stroke`, `median.size`, `median.colour`, `median.color`, `median.fill`, `median.alpha`
Default aesthetics for medians. Set to `sync()` to inherit from the data's aesthetics or to `NULL` to use the data's aesthetics.

`fence.linetype`, `fence.linewidth`, `fence.colour`, `fence.color`, `fence.fill`, `fence.alpha`
Default aesthetics for fences. Set to `sync()` to inherit from the data's aesthetics or to `NULL` to use the data's aesthetics.

`outlier.shape`, `outlier.stroke`, `outlier.size`, `outlier.colour`, `outlier.color`, `outlier.fill`, `outlier.alpha`
Default aesthetics for outliers. Set to `sync()` to inherit from the data's aesthetics or to `NULL` to use the data's aesthetics.

`na.rm` Passed to `ggplot2::layer()`.

`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()`.

Details

geom_bagplot() is designed to pair with stat_bagplot(), analogously to the pairing of ggplot2::geom_boxplot() with ggplot2::stat_boxplot().

Because the optional components are more expensive to compute in this setting, they are controlled by parameters passed to the stat. Auxiliary aesthetics like median.colour are available that override auxiliary defaults, and these in turn override the standard defaults. Auxiliary defaults also take effect when auxiliary aesthetics are passed NULL, so that stat_bagplot() and geom_bagplot() have the same default behavior. Pass sync() (instead of NULL, as in ggplot2::geom_boxplot()) to synchronize an auxiliary aesthetic with its standard counterpart.

WARNING: The trade-off between precision and runtime is greater for depth estimation than for density estimation. At the resolution of the default 100×100 grid, basic examples may vary noticeably when starting from different random seeds.

Value

A ggproto layer.

Aesthetics

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

- x
- y
- component
- linewidth
- linetype
- colour
- fill
- alpha
- shape
- stroke
- size
- group

See Also

Other geom layers: geom_axis(), geom_isoline(), geom_lineranges(), geom_rule(), geom_text_radiate(), geom_vector()

Examples

```
# Motor Trends base plot with factorized cylinder counts
p <- mtcars %>%
  transform(cyl = factor(cyl)) %>%
  ggplot(aes(x = wt, y = disp)) +
```

```
  theme_bw()
# basic bagplot
p + geom_bagplot()
# group by cylinder count
p + geom_bagplot(
  fraction = 0.4, coef = 1.2,
  aes(fill = cyl, linetype = cyl, color = cyl)
)
# using normally unmapped aesthetics
p + geom_bagplot(
  fraction = 0.4, coef = 1.2,
  aes(fill = cyl, linetype = cyl, color = cyl),
  median.color = "black",
  fence.linetype = sync(), fence.colour = "black",
  outlier.shape = "asterisk", outlier.colour = "black"
)
```

---

geom_isoline *Isolines (contour lines)*

Description

geom_isoline() renders isolines along row or column axes.

Usage

```
geom_isoline(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  isoline_text = TRUE,
  by = NULL,
  num = NULL,
  text_dodge = 0.03,
  ...,
  text.size = 3,
  text.angle = 0,
  text.colour = NULL,
  text.color = NULL,
  text.alpha = NULL,
  parse = FALSE,
  check_overlap = FALSE,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by aes(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
stat	The statistical transformation to use on the data for this layer. When using a geom_*() function to construct a layer, the stat argument can be used the override the default coupling between geoms and stats. The stat argument accepts the following: • A Stat ggproto subclass, for example StatCount. • A string naming the stat. To give the stat as a string, strip the function name of the stat_ prefix. For example, to use stat_count(), give the stat as "count". • For more information and other ways to specify the stat, see the layer stat documentation.
position	A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following: • The result of calling a position function, such as position_jitter(). This method allows for passing extra arguments to the position. • A string naming the position adjustment. To give the position as a string, strip the function name of the position_ prefix. For example, to use position_jitter(), give the position as "jitter". • For more information and other ways to specify the position, see the layer position documentation.
isoline_text	Logical; whether to include text value marks along the isolines.
by, num	Intervals between elements or number of elements; specify only one.
text_dodge	Numeric; the orthogonal distance of the text from the axis or isoline, as a proportion of the minimum of the plot width and height.
...	Additional arguments passed to ggplot2::layer().
text.size, text.angle, text.colour, text.color, text.alpha	Default aesthetics for tick mark labels. Set to NULL to inherit from the data's aesthetics.
parse	If TRUE, the labels will be parsed into expressions and displayed as described in ?plotmath.

check_overlap	If TRUE, text that overlaps previous text in the same layer will not be plotted. check_overlap happens at draw time and in the order of the data. Therefore data should be arranged by the label column before calling geom_text(). Note that this argument is not supported by geom_label().
na.rm	Passed to ggplot2::layer().
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().

Details

Isolines are topographical features that separate a plot into regions in which a gradient of interest falls within a specified range. Greenacre (2010) uses them effectively to assist with the projection of markers onto axes.

Value

A ggproto layer.

Aesthetics

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

- x
- y
- colour
- alpha
- linewidth
- linetype
- center, scale
- hjust
- vjust
- family
- fontface
- group

References

Greenacre MJ (2010) *Biplots in Practice*. Fundacion BBVA, ISBN: 978-84-923846. https://www.fbbva.es/microsite/multivariate-statistics/biplots.html

See Also

Other geom layers: geom_axis(), geom_bagplot(), geom_lineranges(), geom_rule(), geom_text_radiate(), geom_vector()

Examples

```
# stack loss gradient
stackloss %>%
  lm(formula = stack.loss ~ Air.Flow + Water.Temp + Acid.Conc.) %>%
  coef() %>%
  as.list() %>% as.data.frame() %>%
  subset(select = c(Air.Flow, Water.Temp, Acid.Conc.)) ->
  coef_data
# isolines along strongest predictors
scale(stackloss, scale = FALSE) %>%
  ggplot(aes(x = Water.Temp, y = Air.Flow)) +
  coord_square() +
  geom_point(aes(size = stack.loss)) + scale_size_area() +
  geom_isoline(data = coef_data)
```

---

geom_lineranges *Intervals depicting ranges, usually about center points*

Description

geom_lineranges() renders horizontal and vertical intervals for a specified subject or variable; geom_pointranges() additionally renders a point at their crosshairs.

Usage

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

geom_pointranges(
  mapping = NULL,
  data = NULL,
  stat = "center",
  position = "identity",
  ...,
  na.rm = FALSE,
```

```
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by aes(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
stat	The statistical transformation to use on the data for this layer. When using a geom_*() function to construct a layer, the stat argument can be used the override the default coupling between geoms and stats. The stat argument accepts the following: • A Stat ggproto subclass, for example StatCount. • A string naming the stat. To give the stat as a string, strip the function name of the stat_ prefix. For example, to use stat_count(), give the stat as "count". • For more information and other ways to specify the stat, see the layer stat documentation.
position	A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following: • The result of calling a position function, such as position_jitter(). This method allows for passing extra arguments to the position. • A string naming the position adjustment. To give the position as a string, strip the function name of the position_ prefix. For example, to use position_jitter(), give the position as "jitter". • For more information and other ways to specify the position, see the layer position documentation.
...	Additional arguments passed to ggplot2::layer().
na.rm	Passed to ggplot2::layer().
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().

Details

The geom_*ranges() layers are shortcuts for equivalently-specified pairs of horizontal and vertical ggplot2::geom_*range() layers. Rather than ggplot2::stat_identity(), they default to stat_center(), so that in practice the summary values do not need to be manually passed.

Value

A ggproto layer.

Aesthetics

geom_lineranges() and geom_pointranges() understand the following aesthetics (required aesthetics are in bold):

- x
- xmin
- xmax
- y
- ymin
- ymax‘
- alpha
- colour
- linewidth
- linetype
- size
- group

See Also

Other geom layers: geom_axis(), geom_bagplot(), geom_isoline(), geom_rule(), geom_text_radiate(), geom_vector()

Examples

```
ggplot(mpg, aes(x = displ, y = hwy, color = drv)) +
  geom_point(alpha = .25) +
  geom_lineranges()

if (require(Hmisc)) {
  ggplot(mpg, aes(x = displ, y = hwy, color = drv)) +
    geom_point(alpha = .25) +
    geom_pointranges(fun.data = mean_sdl, shape = "circle open")
}

mpg %>%
  aggregate(
    x = cbind(displ, hwy) ~ 0,
```

```
  FUN = function(z) c(min = min(z), med = median(z), max = max(z))
) %>%
do.call(what = data.frame) %>%
ggplot(aes(displ.med, hwy.med)) +
geom_pointranges(
  stat = "identity",
  aes(xmin = displ.min, xmax = displ.max, ymin = hwy.min, ymax = hwy.max)
) +
geom_point(data = mpg, aes(displ, hwy), alpha = .5)
```

---

geom_rule *Rulers through or offset from the origin*

Description

geom_rule() renders segments through or orthogonally translated from the origin.

Usage

```
geom_rule(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  axis_labels = TRUE,
  axis_ticks = TRUE,
  axis_text = TRUE,
  by = NULL,
  num = NULL,
  snap_rule = TRUE,
  tick_length = 0.025,
  text_dodge = 0.03,
  label_dodge = 0.03,
  ...,
  axis.colour = NULL,
  axis.color = NULL,
  axis.alpha = NULL,
  label.angle = 0,
  label.colour = NULL,
  label.color = NULL,
  label.alpha = NULL,
  tick.linewidth = 0.25,
  tick.colour = NULL,
  tick.color = NULL,
  tick.alpha = NULL,
  text.size = 2.6,
  text.angle = 0,
  text.hjust = 0.5,
```

```
  text.vjust = 0.5,
  text.family = NULL,
  text.fontface = NULL,
  text.colour = NULL,
  text.color = NULL,
  text.alpha = NULL,
  parse = FALSE,
  check_overlap = FALSE,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by aes(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
stat	The statistical transformation to use on the data for this layer. When using a geom_*() function to construct a layer, the stat argument can be used the override the default coupling between geoms and stats. The stat argument accepts the following: • A Stat ggproto subclass, for example StatCount. • A string naming the stat. To give the stat as a string, strip the function name of the stat_ prefix. For example, to use stat_count(), give the stat as "count". • For more information and other ways to specify the stat, see the layer stat documentation.
position	A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following: • The result of calling a position function, such as position_jitter(). This method allows for passing extra arguments to the position. • A string naming the position adjustment. To give the position as a string, strip the function name of the position_ prefix. For example, to use position_jitter(), give the position as "jitter". • For more information and other ways to specify the position, see the layer position documentation.

`axis_labels`, `axis_ticks`, `axis_text`
: Logical; whether to include labels, tick marks, and text value marks along the axes.

`by`, `num`
: Intervals between elements or number of elements; specify only one.

`snap_rule`
: Logical; whether to snap rule segments to grid values.

`tick_length`
: Numeric; the length of the tick marks, as a proportion of the minimum of the plot width and height.

`text_dodge`
: Numeric; the orthogonal distance of tick mark text from the axis, as a proportion of the minimum of the plot width and height.

`label_dodge`
: Numeric; the orthogonal distance of the axis label from the axis, as a proportion of the minimum of the plot width and height.

`...`
: Additional arguments passed to `ggplot2::layer()`.

`axis.colour`, `axis.color`, `axis.alpha`
: Default aesthetics for axes. Set to NULL to inherit from the data's aesthetics.

`label.angle`, `label.colour`, `label.color`, `label.alpha`
: Default aesthetics for labels. Set to NULL to inherit from the data's aesthetics.

`tick.linewidth`, `tick.colour`, `tick.color`, `tick.alpha`
: Default aesthetics for tick marks. Set to NULL to inherit from the data's aesthetics.

`text.size`, `text.angle`, `text.hjust`, `text.vjust`, `text.family`, `text.fontface`, `text.colour`, `text.color`, `text.alpha`
: Default aesthetics for tick mark labels. Set to NULL to inherit from the data's aesthetics.

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

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

`na.rm`
: Passed to `ggplot2::layer()`.

`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()`.

Details

As implemented here, a rule is just an axis that has a fixed range, usually the limits of the data. `geom_rule()` defaults to `stat = "identity"` to avoid the problem of failing to pass referent data to the referential `stat_rule()`. Therefore, the user must provide the `lower` and `upper` aesthetics, which are used as euclidean lengths in the plotting window. Meanwhile, `stat_rule()` defaults to `geom = "rule"`; see `stat_rule()` for details on this pairing.

Value

A ggproto layer.

Aesthetics

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

- `x`
- `y`
- `lower`
- `upper`
- `yintercept` *or* `xintercept` *or* `xend` and `yend`
- `linetype`
- `linewidth`
- `size`
- `hjust`
- `vjust`
- `colour`
- `alpha`
- `label`
- `family`
- `fontface`
- `center`, `scale`
- `group`

See Also

Other geom layers: `geom_axis()`, `geom_bagplot()`, `geom_isoline()`, `geom_lineranges()`, `geom_text_radiate()`, `geom_vector()`

Examples

```
USJudgeRatings %>%
  subset(select = -c(1, 12)) %>%
  dist(method = "maximum") %>%
  cmdscale() %>%
  as.data.frame() %>%
  setNames(c("PCo1", "PCo2")) %>%
  transform(name = rownames(USJudgeRatings)) ->
  judge_mds
USJudgeRatings %>%
  subset(select = c(CONT, RTEN)) %>%
  setNames(c("contacts", "recommendation")) ->
  judge_meta
lm(as.matrix(judge_meta) ~ as.matrix(judge_mds[, seq(2)])) %>%
```

```
  getElement("coefficients") %>%
  unname() %>% t() %>% as.data.frame() %>%
  setNames(c("Intercept", "PCo1", "PCo2")) %>%
  transform(variable = names(judge_meta)) ->
  judge_lm
ggplot(judge_mds, aes(x = PCo1, y = PCo2)) +
  coord_equal() +
  theme_void() +
  geom_text(aes(label = name), size = 3) +
  stat_rule(
    data = judge_lm, referent = judge_mds,
    aes(center = Intercept, label = variable)
  )
```

`geom_text_radiate` *Text radiating outward from the origin*

Description

`geom_text_radiate()` is adapted from `ggbiplot()` in the off-CRAN extensions of the same name (Vu, 2014; Telford, 2017; Gegzna, 2018). It renders text at specified positions and angles that radiate out from the origin. This layer and its associated ggproto are **deprecated**.

Usage

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

Arguments

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

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

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

`stat` The statistical transformation to use on the data for this layer. When using a `geom_*()` function to construct a layer, the `stat` argument can be used the override the default coupling between geoms and stats. The `stat` argument accepts the following:

- A `Stat` ggproto subclass, for example `StatCount`.
- A string naming the stat. To give the stat as a string, strip the function name of the `stat_` prefix. For example, to use `stat_count()`, give the stat as `"count"`.
- For more information and other ways to specify the stat, see the layer stat documentation.

`position` A position adjustment to use on the data for this layer. Cannot be jointy specified with `nudge_x` or `nudge_y`. This can be used in various ways, including to prevent overplotting and improving the display. The `position` argument accepts the following:

- The result of calling a position function, such as `position_jitter()`.
- A string nameing the position adjustment. To give the position as a string, strip the function name of the `position_` prefix. For example, to use `position_jitter()`, give the position as `"jitter"`.
- For more information and other ways to specify the position, see the layer position documentation.

`...` Other arguments passed on to `layer()`'s `params` argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the `position` argument, or aesthetics that are required can *not* be passed through `....` Unknown arguments that are not part of the 4 categories below are ignored.

- Static aesthetics that are not mapped to a scale, but are at a fixed value and apply to the layer as a whole. For example, `colour = "red"` or `linewidth = 3`. The geom's documentation has an **Aesthetics** section that lists the available options. The 'required' aesthetics cannot be passed on to the `params`. Please note that while passing unmapped aesthetics as vectors is technically possible, the order and required length is not guaranteed to be parallel to the input data.
- When constructing a layer using a `stat_*()` function, the `...` argument can be used to pass on parameters to the `geom` part of the layer. An example of this is `stat_density(geom = "area", outline.type = "both")`. The geom's documentation lists which parameters it can accept.
- Inversely, when constructing a layer using a `geom_*()` function, the `...` argument can be used to pass on parameters to the `stat` part of the layer. An example of this is `geom_area(stat = "density", adjust = 0.5)`. The stat's documentation lists which parameters it can accept.

- The key_glyph argument of layer() may also be passed on through .... This can be one of the functions described as key glyphs, to change the display of the layer in the legend.

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

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

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().

Value

A ggproto layer.

Aesthetics

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

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

References

Vincent Q. Vu (2014). ggbiplot: A 'ggplot2' based biplot. R package version 0.55. https://github.com/vqv/ggbiplot, experimental branch

Richard J Telford (2017). ggbiplot: A 'ggplot2' based biplot. R package version 0.6. https://github.com/richardjtelford/ggbiplot (fork), experimental branch

Vilmantas Gegzna (2018). ggbiplot: A 'ggplot2' based biplot. R package version 0.55. https://github.com/forked-packages/ggbiplot (fork), experimental branch

See Also

Other geom layers: geom_axis(), geom_bagplot(), geom_isoline(), geom_lineranges(), geom_rule(), geom_vector()

`geom_vector` *Vectors from the origin*

Description

`geom_vector()` renders arrows from the origin to points, optionally with text radiating outward.

Usage

```
geom_vector(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  arrow = default_arrow,
  lineend = "round",
  linejoin = "mitre",
  vector_labels = TRUE,
  ...,
  label.colour = NULL,
  label.color = NULL,
  label.alpha = NULL,
  parse = FALSE,
  check_overlap = FALSE,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by aes(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
stat	The statistical transformation to use on the data for this layer. When using a geom_*() function to construct a layer, the stat argument can be used the override the default coupling between geoms and stats. The stat argument accepts the following: • A Stat ggproto subclass, for example StatCount. • A string naming the stat. To give the stat as a string, strip the function name of the stat_ prefix. For example, to use stat_count(), give the stat as "count". • For more information and other ways to specify the stat, see the layer stat documentation.
position	A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following: • The result of calling a position function, such as position_jitter(). This method allows for passing extra arguments to the position. • A string naming the position adjustment. To give the position as a string, strip the function name of the position_ prefix. For example, to use position_jitter(), give the position as "jitter". • For more information and other ways to specify the position, see the layer position documentation.
arrow	Specification for arrows, as created by grid::arrow(), or else NULL for no arrows.
lineend	Line end style (round, butt, square).
linejoin	Line join style (round, mitre, bevel).
vector_labels	Logical; whether to include labels radiating outward from the vectors.
...	Additional arguments passed to ggplot2::layer().
label.colour, label.color, label.alpha	Default aesthetics for labels. Set to NULL to inherit from the data's aesthetics.
parse	If TRUE, the labels will be parsed into expressions and displayed as described in ?plotmath.

check_overlap	If TRUE, text that overlaps previous text in the same layer will not be plotted. check_overlap happens at draw time and in the order of the data. Therefore data should be arranged by the label column before calling geom_text(). Note that this argument is not supported by geom_label().
na.rm	Passed to ggplot2::layer().
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().

Details

Vectors are positions relative to some common reference point, in this case the origin; they comprise direction and magnitude. Vectors are usually represented with arrows rather than markers (points).

Vectors are commonly used to represent numerical variables in biplots, as by Gabriel (1971) and Greenacre (2010). Gardner & le Roux (2002) refer to these as Gabriel biplots. This layer, with optional radiating text labels, is adapted from ggbiplot() in the off-CRAN extensions of the same name (Vu, 2014; Telford, 2017; Gegzna, 2018).

Value

A ggproto layer.

Aesthetics

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

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

References

Gabriel KR (1971) "The biplot graphic display of matrices with application to principal component analysis". *Biometrika* 58(3), 453–467. doi:10.1093/biomet/58.3.453

Greenacre MJ (2010) *Biplots in Practice*. Fundacion BBVA, ISBN: 978-84-923846. https://www.fbbva.es/microsite/multivariate-statistics/biplots.html

Gardner S, le Roux N (2002) "Biplot Methodology for Discriminant Analysis Based upon Robust Methods and Principal Curves". *Classification, Clustering, and Data Analysis: Recent Advances and Applications*: 169–176. https://link.springer.com/chapter/10.1007/978-3-642-56181-8_18

Vincent Q. Vu (2014). ggbiplot: A 'ggplot2' based biplot. R package version 0.55. https://github.com/vqv/ggbiplot, `experimental` branch

Richard J Telford (2017). ggbiplot: A 'ggplot2' based biplot. R package version 0.6. https://github.com/richardjtelford/ggbiplot (fork), `experimental` branch

Vilmantas Gegzna (2018). ggbiplot: A 'ggplot2' based biplot. R package version 0.55. https://github.com/forked-packages/ggbiplot (fork), `experimental` branch

See Also

Other geom layers: `geom_axis()`, `geom_bagplot()`, `geom_isoline()`, `geom_lineranges()`, `geom_rule()`, `geom_text_radiate()`

Examples

```
# multidimensional scaling of covariances
ability.cov$cov %>%
  cov2cor() %>%
  eigen() %>% getElement("vectors") %>%
  as.data.frame() %>%
  transform(test = rownames(ability.cov$cov)) ->
  ability_cor_eigen
ability_cor_eigen %>%
  ggplot(aes(-V1, V2, label = test)) +
  coord_square() + theme_void() +
  geom_vector(check_overlap = TRUE) +
  scale_y_continuous(expand = expansion(mult = .2)) +
  ggtitle("Ability and intelligence test covariances")
# multidimensional scaling of correlations
ability.cov$cov %>%
  eigen() %>% getElement("vectors") %>%
  as.data.frame() %>%
  transform(test = rownames(ability.cov$cov)) ->
  ability_cor_eigen
ability_cor_eigen %>%
  ggplot(aes(-V1, -V2, label = test)) +
  coord_square() + theme_void() +
  geom_vector(check_overlap = TRUE) +
  expand_limits(x = c(-1, 1), y = c(-1, 1)) +
  ggtitle("Ability and intelligence test covariances")
```

gggda-ggproto *ggproto classes created and adapted for gggda*

Description

gggda introduces several ggproto classes for coordinate systems, statistical transformations, and geometric constructions specific to multivariate analysis or following geometric data analysis principles.

Details

The new ggprotos are inspired by two entangled but distinct threads in multivariate data visualization: First, several geometric constructions have been proposed to generalize both numeric and graphical summaries of univariate data to the bivariate setting. Among these are various "peeling" procedures like that by successive convex hulls, which generalize the concept of rank (Green, 1981); and the depth-based bag-and-bolster plot, designed as a bivariate analog of the box-and-whisker plot (Rousseeuw &al, 1999). Second, the use of biplots to visualize singular value–decomposed data benefits from being able to encode variables with different graphical elements than the markers used to encode cases—for example, vectors (arrows emanating from the origin; Gabriel, 1971), calibrated axes (Gower & Hand, 1996), and prediction regions (Gardner & le Roux, 2002).

References

Green PJ (1981) "Peeling Bivariate Data". *Interpreting Multivariate Data* Chapter 1, 3–19. John Wiley & Sons, Ltd, ISBN 978-0-471-28039-2.

Rousseeuw PJ, Ruts I, & Tukey JW (1999) "The Bagplot: A Bivariate Boxplot". *The American Statistician*, **53**(4): 382–387. doi:10.1080/00031305.1999.10474494

Gabriel KR (1971) "The biplot graphic display of matrices with application to principal component analysis". *Biometrika* 58(3), 453–467. doi:10.1093/biomet/58.3.453

Gower JC & Hand DJ (1996) *Biplots*. Chapman & Hall, ISBN: 0-412-71630-5.

Gardner S, le Roux N (2002) "Biplot Methodology for Discriminant Analysis Based upon Robust Methods and Principal Curves". *Classification, Clustering, and Data Analysis: Recent Advances and Applications*: 169–176. https://link.springer.com/chapter/10.1007/978-3-642-56181-8_18

See Also

ggplot2::ggplot2-ggproto and ggplot2::ggproto for explanations of base ggproto classes in **ggplot2** and how to create new ones.

peel_hulls *Bivariate data peelings*

Description

Use convex hulls (and eventually other peelings) to order bivariate data.

Usage

```
peel_hulls(
  x,
  y = NULL,
  num = NULL,
  by = 1L,
  breaks = c(0.5),
  cut = c("above", "below")
)
```

Arguments

x, y	coordinate vectors of points. This can be specified as two vectors x and y, a 2-column matrix x, a list x with two components, etc, see xy.coords.
num	A positive integer; the number of hulls to peel. Pass Inf for all hulls.
by	A positive integer; with what frequency to include consecutive hulls, pairs with num.
breaks	A numeric vector of fractions (between 0 and 1) of the data to contain in each hull; overridden by num.
cut	Character; one of "above" and "below", indicating whether each hull should contain at least or at most breaks of the data, respectively.

Details

Methods for peeling bivariate data into concentric tiers generalize the univariate concept of rank to separate core versus peripheral cases (Green, 1981).

The code for peeling convex hulls was adapted from plothulls() in the **aplpack** package. Other peeling options should be implemented soon.

Value

A matrix with some or all of the following columns:

x,y original coordinates

i position in input matrix or vectors

hull index of hull, starting from outermost

frac value of breaks used to determine hull

prop proportion of data within hull

References

Green PJ (1981) "Peeling Bivariate Data". *Interpreting Multivariate Data* Chapter 1, 3–19. John Wiley & Sons, Ltd, ISBN 978-0-471-28039-2.

Examples

```
x <- mtcars$disp; y <- mtcars$mpg

# all hulls
peel_hulls(x, y, num = Inf)

# every third hull
peel_hulls(x, y, num = Inf, by = 3)

# tertile hulls, cut below
peel_hulls(x, y, breaks = seq(0, 1, length.out = 4))

# tertile hulls, cut above
peel_hulls(x, y, breaks = seq(0, 1, length.out = 4), cut = "below")
```

---

stat_bagplot *Bagplots*

Description

Construct medians, bags, fences, and outlier specifications for bagplots.

Usage

```
stat_bagplot(
  mapping = NULL,
  data = NULL,
  geom = "bagplot",
  position = "identity",
  fraction = 0.5,
  coef = 3,
  median = TRUE,
  fence = TRUE,
  outliers = TRUE,
  show.legend = NA,
  inherit.aes = TRUE,
  ...
)
```

Arguments

mapping	Set of aesthetic mappings created by aes(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
geom	The geometric object to use to display the data for this layer. When using a stat_*() function to construct a layer, the geom argument can be used to override the default coupling between stats and geoms. The geom argument accepts the following: • A Geom ggproto subclass, for example GeomPoint. • A string naming the geom. To give the geom as a string, strip the function name of the geom_ prefix. For example, to use geom_point(), give the geom as "point". • For more information and other ways to specify the geom, see the layer geom documentation.
position	A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following: • The result of calling a position function, such as position_jitter(). This method allows for passing extra arguments to the position. • A string naming the position adjustment. To give the position as a string, strip the function name of the position_ prefix. For example, to use position_jitter(), give the position as "jitter". • For more information and other ways to specify the position, see the layer position documentation.
fraction	Fraction of the data to include in the bag.
coef	Scale factor of the fence relative to the bag.
median, fence, outliers	Logical indicators whether to include median, fence, and outliers in the composite output.
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().

... Arguments passed on to stat_depth

notion Character; the name of the depth function (passed to ddalpha::depth.()).

notion_params List of additional parameters passed via ... to ddalpha::depth.().

Details

A bagplot comprises a single, often filled, depth contour (the "bag") overlaid with the hull of its union with the data points contained in its scaled expansion from the depth median (the "fence") and a scatterplot of outliers beyond the fence (the "loop"). Rousseeuw &al (1999) suggest the term "bag-and-bolster plot" evocative of the "box-and-whisker plot".

While the depth median can be obtained using stat_center(), the data depth values used to compute it are also used to demarcate the bag, so it is implemented separately in StatBagplot$compute_group() for efficiency.

stat_bagplot() is designed to pair with geom_bagplot(), analogously to the pairing of ggplot2::stat_boxplot() with ggplot2::geom_boxplot(). In particular, GeomBagplot is the only ggproto that recognizes the computed variable component, used by StatBagplot to separate data for the four bagplot elements.

Value

A ggproto layer.

Multidimensional position aesthetics

This statistical transformation is compatible with the convenience function aes_coord().

Some transformations (e.g. stat_center()) commute with projection to the lower (1 or 2)-dimensional biplot space. If they detect aesthetics of the form ..coord[0-9]+, then ..coord1 and ..coord2 are converted to x and y while any remaining are ignored.

Other transformations (e.g. stat_spantree()) yield different results in a lower-dimensional biplot when they are computed before versus after projection. If the stat layer detects these aesthetics, then the transformation is performed before projection, and the results in the first two dimensions are returned as x and y.

A small number of transformations (stat_rule()) are incompatible with these aesthetics but will accept aes_coord() without warning.

Computed variables

These are calculated during the statistical transformation and can be accessed with delayed evaluation.

component the component of the composite plot; used internally

References

Rousseeuw PJ, Ruts I, & Tukey JW (1999) "The Bagplot: A Bivariate Boxplot". *The American Statistician*, **53**(4): 382–387. doi:10.1080/00031305.1999.10474494

See Also

Other stat layers: stat_center(), stat_chull(), stat_cone(), stat_depth(), stat_rule(), stat_scale(), stat_spantree()

Examples

```
# petroleum rock base plot
p <- ggplot(rock, aes(area, shape, size = peri)) + theme_bw()
# scatterplot
p + geom_point()
# NB: Non-standard aesthetics are handled as in version > 3.5.1; see:
# https://github.com/tidyverse/ggplot2/issues/6191
# custom bag fraction, coefficient, and aesthetics
p + stat_bagplot(fraction = .4, coef = 1.5,
                 outlier_gp = list(shape = "asterisk"))
# invisible fence
p + stat_bagplot(fence = FALSE)
```

---

stat_center *Centers and spreads for bivariate data*

Description

Centers and spreads for bivariate data

Usage

```
stat_center(
  mapping = NULL,
  data = NULL,
  geom = "point",
  position = "identity",
  show.legend = NA,
  inherit.aes = TRUE,
  ...,
  fun.data = NULL,
  fun = NULL,
  fun.center = NULL,
  fun.min = NULL,
  fun.max = NULL,
  fun.ord = NULL,
  fun.args = list()
)

stat_star(
  mapping = NULL,
  data = NULL,
  geom = "segment",
```

```
  position = "identity",
  show.legend = NA,
  inherit.aes = TRUE,
  ...,
  fun.data = NULL,
  fun = NULL,
  fun.center = NULL,
  fun.ord = NULL,
  fun.args = list()
)
```

Arguments

mapping	Set of aesthetic mappings created by aes(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
geom	The geometric object to use to display the data for this layer. When using a stat_*() function to construct a layer, the geom argument can be used to override the default coupling between stats and geoms. The geom argument accepts the following: • A Geom ggproto subclass, for example GeomPoint. • A string naming the geom. To give the geom as a string, strip the function name of the geom_ prefix. For example, to use geom_point(), give the geom as "point". • For more information and other ways to specify the geom, see the layer geom documentation.
position	A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following: • The result of calling a position function, such as position_jitter(). This method allows for passing extra arguments to the position. • A string naming the position adjustment. To give the position as a string, strip the function name of the position_ prefix. For example, to use position_jitter(), give the position as "jitter". • For more information and other ways to specify the position, see the layer position documentation.

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().

... — Additional arguments passed to ggplot2::layer().

fun.data — A function that is given the complete data and should return a data frame with variables ymin, y, and ymax.

fun.center — Deprecated alias to fun.

fun.min, fun, fun.max — Alternatively, supply three individual functions that are each passed a vector of values and should return a single number.

fun.ord — Alternatively to the ggplot2::stat_summary_bin() parameters, supply a summary function that takes a matrix as input and returns a named column summary vector. Overridden by fun.data and fun, cannot be used together with fun.min and fun.max.

fun.args — Optional additional arguments passed on to the functions.

Value

A ggproto layer.

Multidimensional position aesthetics

This statistical transformation is compatible with the convenience function aes_coord().

Some transformations (e.g. stat_center()) commute with projection to the lower (1 or 2)-dimensional biplot space. If they detect aesthetics of the form ..coord[0-9]+, then ..coord1 and ..coord2 are converted to x and y while any remaining are ignored.

Other transformations (e.g. stat_spantree()) yield different results in a lower-dimensional biplot when they are computed before versus after projection. If the stat layer detects these aesthetics, then the transformation is performed before projection, and the results in the first two dimensions are returned as x and y.

A small number of transformations (stat_rule()) are incompatible with these aesthetics but will accept aes_coord() without warning.

Computed variables

These are calculated during the statistical transformation and can be accessed with delayed evaluation.

xmin,ymin,xmax,ymax results of fun.min,fun.max applied to x,y

See Also

Other stat layers: stat_bagplot(), stat_chull(), stat_cone(), stat_depth(), stat_rule(), stat_scale(), stat_spantree()

Examples

```
ggplot(mpg, aes(x = displ, y = cty, shape = drv)) +
  geom_point() +
  stat_center(fun = "median", size = 5, alpha = .5)

ggplot(mpg, aes(x = displ, y = cty, shape = drv, linetype = drv)) +
  stat_center(size = 3) +
  stat_star()
```

---

stat_chull *Convex hulls and hull peelings*

Description

Restrict planar data to the boundary points of its convex hull, or of nested convex hulls containing specified fractions of points.

Usage

```
stat_chull(
  mapping = NULL,
  data = NULL,
  geom = "polygon",
  position = "identity",
  show.legend = NA,
  inherit.aes = TRUE,
  ...
)

stat_peel(
  mapping = NULL,
  data = NULL,
  geom = "polygon",
  position = "identity",
  num = NULL,
  by = 1L,
  breaks = c(0.5),
  cut = c("above", "below"),
  show.legend = NA,
  inherit.aes = TRUE,
  ...
)
```

Arguments

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

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

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

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

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

geom — The geometric object to use to display the data for this layer. When using a stat_*() function to construct a layer, the geom argument can be used to override the default coupling between stats and geoms. The geom argument accepts the following:

- A Geom ggproto subclass, for example GeomPoint.
- A string naming the geom. To give the geom as a string, strip the function name of the geom_ prefix. For example, to use geom_point(), give the geom as "point".
- For more information and other ways to specify the geom, see the layer geom documentation.

position — A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following:

- The result of calling a position function, such as position_jitter(). This method allows for passing extra arguments to the position.
- A string naming the position adjustment. To give the position as a string, strip the function name of the position_ prefix. For example, to use position_jitter(), give the position as "jitter".
- For more information and other ways to specify the position, see the layer position documentation.

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().

... — Additional arguments passed to ggplot2::layer().

num — A positive integer; the number of hulls to peel. Pass Inf for all hulls.

by — A positive integer; with what frequency to include consecutive hulls, pairs with num.

breaks — A numeric vector of fractions (between 0 and 1) of the data to contain in each hull; overridden by num.

cut — Character; one of "above" and "below", indicating whether each hull should contain at least or at most breaks of the data, respectively.

Details

As used in a **ggplot2** vignette, `stat_chull()` restricts a dataset with `x` and `y` variables to the points that lie on its convex hull.

Building on this, `stat_peel()` returns hulls from a *convex hull peeling*: a subset of sequentially removed hulls containing specified fractions of the data.

Value

A ggproto layer.

Multidimensional position aesthetics

This statistical transformation is compatible with the convenience function `aes_coord()`.

Some transformations (e.g. `stat_center()`) commute with projection to the lower (1 or 2)-dimensional biplot space. If they detect aesthetics of the form `..coord[0-9]+`, then `..coord1` and `..coord2` are converted to `x` and `y` while any remaining are ignored.

Other transformations (e.g. `stat_spantree()`) yield different results in a lower-dimensional biplot when they are computed before versus after projection. If the stat layer detects these aesthetics, then the transformation is performed before projection, and the results in the first two dimensions are returned as `x` and `y`.

A small number of transformations (`stat_rule()`) are incompatible with these aesthetics but will accept `aes_coord()` without warning.

Computed variables

These are calculated during the statistical transformation and can be accessed with delayed evaluation.

`hull` the position of `breaks` that defines each hull

`frac` the value of `breaks` that defines each hull

`prop` the actual proportion of data within each hull

References

Barnett V (1976) "The Ordering of Multivariate Data". *Journal of the Royal Statistical Society: Series A (General)*, **139**(3): 318–344. doi:10.2307/2344839

See Also

Other stat layers: `stat_bagplot()`, `stat_center()`, `stat_cone()`, `stat_depth()`, `stat_rule()`, `stat_scale()`, `stat_spantree()`

Examples

```
ggplot(USJudgeRatings, aes(x = INTG, y = PREP)) +
  geom_point() +
  stat_chull(alpha = .5)
ggplot(USJudgeRatings, aes(x = INTG, y = PREP)) +
```

```
  stat_peel(
    aes(alpha = after_stat(hull)),
    breaks = seq(.1, .9, .2)
  )
ggplot(USJudgeRatings, aes(x = INTG, y = PREP)) +
  stat_peel(
    aes(alpha = after_stat(hull)),
    num = 6, by = 2, color = "black"
  )

# specify fractions of points
ggplot(iris, aes(Sepal.Width, Sepal.Length, color = Species)) +
  stat_peel(aes(fill = Species, alpha = after_stat(frac)),
            breaks = seq(.1, .9, .2)) +
  scale_alpha_continuous(trans = scales::reverse_trans()) +
  geom_point()
# specify number of peels
ggplot(iris, aes(Sepal.Width, Sepal.Length, color = Species)) +
  stat_peel(fill = "transparent", num = 3) +
  geom_point()
# mapping to opacity overrides transparency
ggplot(iris, aes(Sepal.Width, Sepal.Length, color = Species)) +
  stat_peel(aes(alpha = after_stat(hull)), fill = "transparent", num = 3) +
  geom_point()
```

`stat_cone` *Conical hull*

Description

Restrict planar data to the points that lie on its conical hull.

Usage

```
stat_cone(
  mapping = NULL,
  data = NULL,
  geom = "path",
  position = "identity",
  origin = FALSE,
  show.legend = NA,
  inherit.aes = TRUE,
  ...
)
```

Arguments

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

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

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

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

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

`geom` The geometric object to use to display the data for this layer. When using a `stat_*()` function to construct a layer, the `geom` argument can be used to override the default coupling between stats and geoms. The `geom` argument accepts the following:

- A `Geom` ggproto subclass, for example `GeomPoint`.
- A string naming the geom. To give the geom as a string, strip the function name of the `geom_` prefix. For example, to use `geom_point()`, give the geom as `"point"`.
- For more information and other ways to specify the geom, see the layer geom documentation.

`position` A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The `position` argument accepts the following:

- The result of calling a position function, such as `position_jitter()`. This method allows for passing extra arguments to the position.
- A string naming the position adjustment. To give the position as a string, strip the function name of the `position_` prefix. For example, to use `position_jitter()`, give the position as `"jitter"`.
- For more information and other ways to specify the position, see the layer position documentation.

`origin` Logical; whether to include the origin with the transformed data. Defaults to `FALSE`.

`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()`.

`...` Additional arguments passed to `ggplot2::layer()`.

Value

A ggproto layer.

Multidimensional position aesthetics

This statistical transformation is compatible with the convenience function `aes_coord()`.

Some transformations (e.g. `stat_center()`) commute with projection to the lower (1 or 2)-dimensional biplot space. If they detect aesthetics of the form `..coord[0-9]+`, then `..coord1` and `..coord2` are converted to `x` and `y` while any remaining are ignored.

Other transformations (e.g. `stat_spantree()`) yield different results in a lower-dimensional biplot when they are computed before versus after projection. If the stat layer detects these aesthetics, then the transformation is performed before projection, and the results in the first two dimensions are returned as `x` and `y`.

A small number of transformations (`stat_rule()`) are incompatible with these aesthetics but will accept `aes_coord()` without warning.

See Also

Other stat layers: `stat_bagplot()`, `stat_center()`, `stat_chull()`, `stat_depth()`, `stat_rule()`, `stat_scale()`, `stat_spantree()`

Examples

```
mtcars$name <- rownames(mtcars)
ggplot(mtcars, aes(wt, mpg, label = name)) +
  geom_text(size = 3) +
  stat_cone()
ggplot(mtcars, aes(hp, mpg, label = name)) +
  geom_text(size = 3) +
  stat_cone(origin = TRUE, linetype = "dotted")
```

---

stat_depth *Depth estimates and contours*

Description

Estimate data depth using `ddalpha::depth.()`.

Usage

```
stat_depth(
  mapping = NULL,
  data = NULL,
  geom = "contour",
  position = "identity",
  contour = TRUE,
  contour_var = "depth",
  notion = "zonoid",
  notion_params = list(),
  n = 100L,
```

```
  show.legend = NA,
  inherit.aes = TRUE,
  ...
)

stat_depth_filled(
  mapping = NULL,
  data = NULL,
  geom = "contour_filled",
  position = "identity",
  contour = TRUE,
  contour_var = "depth",
  notion = "zonoid",
  notion_params = list(),
  n = 100L,
  show.legend = NA,
  inherit.aes = TRUE,
  ...
)
```

Arguments

mapping	Set of aesthetic mappings created by aes(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
geom	The geometric object to use to display the data for this layer. When using a stat_*() function to construct a layer, the geom argument can be used to override the default coupling between stats and geoms. The geom argument accepts the following: • A Geom ggproto subclass, for example GeomPoint. • A string naming the geom. To give the geom as a string, strip the function name of the geom_ prefix. For example, to use geom_point(), give the geom as "point". • For more information and other ways to specify the geom, see the layer geom documentation.
position	A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following:

- The result of calling a position function, such as `position_jitter()`. This method allows for passing extra arguments to the position.
- A string naming the position adjustment. To give the position as a string, strip the function name of the `position_` prefix. For example, to use `position_jitter()`, give the position as `"jitter"`.
- For more information and other ways to specify the position, see the layer position documentation.

`contour` If `TRUE`, contour the results of the depth estimation.

`contour_var` Character string identifying the variable to contour by. Can be one of `"depth"` or `"ndepth"`. See the section on computed variables for details.

`notion` Character; the name of the depth function (passed to `ddalpha::depth.()`).

`notion_params` List of additional parameters passed via `...` to `ddalpha::depth.()`.

`n` Number of grid points in each direction.

`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()`.

`...` Arguments passed on to `ggplot2::geom_contour`

`bins` Number of contour bins. Overridden by `breaks`.

`binwidth` The width of the contour bins. Overridden by `bins`.

`breaks` One of:

- Numeric vector to set the contour breaks
- A function that takes the range of the data and binwidth as input and returns breaks as output. A function can be created from a formula (e.g. ~ fullseq(.x, .y)).

Overrides `binwidth` and `bins`. By default, this is a vector of length ten with `pretty()` breaks.

Details

Depth is an extension of the univariate notion of rank to bivariate (and sometimes multivariate) data (Rousseeuw &al, 1999). It comes in several flavors and is the basis for bagplots.

`stat_depth()` is adapted from `ggplot2::stat_density_2d()` and returns depth values over a grid in the same format, so it is neatly paired with `ggplot2::geom_contour()`.

Value

A ggproto layer.

Multidimensional position aesthetics

This statistical transformation is compatible with the convenience function `aes_coord()`.

Some transformations (e.g. `stat_center()`) commute with projection to the lower (1 or 2)-dimensional biplot space. If they detect aesthetics of the form `..coord[0-9]+`, then `..coord1` and `..coord2` are converted to `x` and `y` while any remaining are ignored.

Other transformations (e.g. `stat_spantree()`) yield different results in a lower-dimensional biplot when they are computed before versus after projection. If the stat layer detects these aesthetics, then the transformation is performed before projection, and the results in the first two dimensions are returned as `x` and `y`.

A small number of transformations (`stat_rule()`) are incompatible with these aesthetics but will accept `aes_coord()` without warning.

Computed variables

These are calculated during the statistical transformation and can be accessed with delayed evaluation.

`stat_depth()` and `stat_depth_filled()` compute different variables depending on whether contouring is turned on or off. With contouring off (`contour = FALSE`), both stats behave the same, and the following variables are provided:

`depth` the depth estimate

`ndepth` depth estimate, scaled to a maximum of 1

With contouring on (`contour = TRUE`), either `ggplot2::stat_contour()` or `ggplot2::stat_contour_filled()` is run after the depth estimate has been obtained, and the computed variables are determined by these stats.

References

Rousseeuw PJ, Ruts I, & Tukey JW (1999) "The Bagplot: A Bivariate Boxplot". *The American Statistician*, **53**(4): 382–387. doi:10.1080/00031305.1999.10474494

See Also

Other stat layers: `stat_bagplot()`, `stat_center()`, `stat_chull()`, `stat_cone()`, `stat_rule()`, `stat_scale()`, `stat_spantree()`

Examples

```
# base Motor Trends plot
b <- ggplot(mtcars, aes(wt, disp)) + geom_point()

# depth raster
b + geom_raster(stat = "depth", aes(fill = after_stat(depth)))
# depth grid
b + stat_depth(
  geom = "point", contour = FALSE,
  aes(size = after_stat(depth)), n = 20
)
```

```
# depth contours
b + geom_contour(stat = "depth", contour = TRUE)
# depth bands
b + geom_contour_filled(stat = "depth_filled", contour = TRUE, alpha = .75)
# contours colored by group
b + stat_depth(aes(color = factor(cyl)))
# custom depth notion
b + stat_depth(
  aes(color = factor(cyl)),
  notion = "halfspace", notion_params = list(exact = TRUE)
)

# contours faceted by group
b + stat_depth_filled(alpha = .75) +
  facet_wrap(facets = vars(factor(cyl)))
# scaled to the unit interval
b + stat_depth_filled(contour_var = "ndepth", alpha = .75) +
  facet_wrap(facets = vars(factor(cyl)))
```

`stat_referent` *Transformations with respect to reference data*

Description

Compute statistics with respect to a reference data set with shared positional variables.

Usage

```
stat_referent(
  mapping = NULL,
  data = NULL,
  geom = "blank",
  position = "identity",
  referent = NULL,
  show.legend = NA,
  inherit.aes = TRUE,
  ...
)

## S3 method for class 'LayerRef'
ggplot_add(object, plot, ...)
```

Arguments

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

data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
geom	The geometric object to use to display the data for this layer. When using a stat_*() function to construct a layer, the geom argument can be used to override the default coupling between stats and geoms. The geom argument accepts the following: • A Geom ggproto subclass, for example GeomPoint. • A string naming the geom. To give the geom as a string, strip the function name of the geom_ prefix. For example, to use geom_point(), give the geom as "point". • For more information and other ways to specify the geom, see the layer geom documentation.
position	A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following: • The result of calling a position function, such as position_jitter(). This method allows for passing extra arguments to the position. • A string naming the position adjustment. To give the position as a string, strip the function name of the position_ prefix. For example, to use position_jitter(), give the position as "jitter". • For more information and other ways to specify the position, see the layer position documentation.
referent	The reference data set; see Details.
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().
...	Additional arguments passed to ggplot2::layer().
object	An object to add to the plot
plot	The ggplot object to add object to

Details

Often in geometric data analysis a statistical transformation applied to data X will also depend on data Y, for example when drawing the projections of vectors X onto vectors Y. The stat layer

`stat_referent()` accepts Y as an argument to the `referent` parameter and pre-processes them using the existing positional aesthetic mappings to `x` and `y`.

The ggproto can be used as a parent to more elaborate statistical transformations, or the stat can be paired with geoms that expect the `referent` parameter and use it to position their transformations of X. It pairs by default to `[ggplot2::geom_blank()]` so as to prevent possibly confusing output.

Value

A ggproto layer.

Examples

```
# simplify the Motor Trends data to two predictors legible at aspect ratio 1
mtcars %>%
  transform(hp00 = hp/100) %>%
  subset(select = c(mpg, hp00, wt)) ->
  subcars
# compute the gradient of `mpg` against these two predictors
lm(mpg ~ hp00 + wt, subcars) %>%
  coefficients() %>%
  as.list() %>% as.data.frame() ->
  grad
# use the gradient as a reference (to no effect in this basic ggproto)
ggplot(subcars, aes(x = hp00, y = wt)) +
  coord_equal() +
  geom_point() +
  stat_referent(referent = grad)
ggplot(subcars, aes(x = hp00, y = wt)) +
  coord_equal() +
  stat_referent(geom = "point", referent = grad)
```

---

stat_rule *Construct limited rules offset from the origin*

Description

Determine axis limits and offset vectors from reference data.

Usage

```
stat_rule(
  mapping = NULL,
  data = NULL,
  geom = "rule",
  position = "identity",
  fun.lower = "minpp",
  fun.upper = "maxpp",
  fun.offset = "minabspp",
```

```
  fun.args = list(),
  referent = NULL,
  show.legend = NA,
  inherit.aes = TRUE,
  ...
)

minpp(x, p = 0.1)

maxpp(x, p = 0.1)

minabspp(x, p = 0.1)
```

Arguments

mapping	Set of aesthetic mappings created by aes(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
geom	The geometric object to use to display the data for this layer. When using a stat_*() function to construct a layer, the geom argument can be used to override the default coupling between stats and geoms. The geom argument accepts the following: • A Geom ggproto subclass, for example GeomPoint. • A string naming the geom. To give the geom as a string, strip the function name of the geom_ prefix. For example, to use geom_point(), give the geom as "point". • For more information and other ways to specify the geom, see the layer geom documentation.
position	A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following: • The result of calling a position function, such as position_jitter(). This method allows for passing extra arguments to the position. • A string naming the position adjustment. To give the position as a string, strip the function name of the position_ prefix. For example, to use position_jitter(), give the position as "jitter". • For more information and other ways to specify the position, see the layer position documentation.

`fun.lower`, `fun.upper`, `fun.offset`
: Functions used to determine the limits of the rules and the translations of the axes from the projections of `referent` onto the axes and onto their normal vectors.

`fun.args`
: Optional additional arguments passed on to the functions.

`referent`
: The reference data set; see Details.

`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()`.

`...`
: Additional arguments passed to `ggplot2::layer()`.

`x`
: A numeric vector.

`p`
: A numeric value; the proportion of a range used as a buffer.

Details

Biplots with several axes can become cluttered and illegible. When this happens, Gower, Gardner–Lubbe, & le Roux (2011) recommend to translate the axes to a new point of intersection away from the origin, adjusting the axis markers accordingly. Then the axes converge in a region of the plot offset from most position markers or other elements. An alternative solution, implemented in the **bipl5** package (`https://github.com/RuanBuys/bipl5`), is to translate each axis orthogonally away from the origin, which preserves the axis markers. This is the technique implemented here.

Separately, axes that fill the plotting window are uninformative when they exceed the range of the plotted position markers projected onto them. They may even be misinformative, suggesting that linear relationships extrapolate outside the data range. In these cases, Gower and Harding (1988) recommend using finite ranges determined by the data projection onto each axis.

Three functions control these operations: `fun.offset` computes the orthogonal distance of each axis from the origin, and `fun.lower` and `fun.upper` compute the distance along each axis of the endpoints to the (offset) origin. Both functions depend on what position data is to be offset from or limited to, which must be passed manually to the `referent` parameter.

Value

A ggproto layer.

Referential stats

This statistical transformation is done with respect to reference data passed to `referent` (ignored if `NULL`, the default, possibly resulting in empty output). See `stat_referent()` for more details. This relies on a sleight of hand through a new undocumented `LayerRef` class and associated `ggplot2::ggplot_add()` method. As a result, only layers constructed using this `stat_*()` shortcut will pass the necessary positional aesthetics to the `$setup_params()` step, making them available to pre-process `referent` data.

The biplot shortcuts automatically substitute the complementary matrix factor for `referent = NULL` and will use an integer vector to select a subset from this factor. These uses do not require the mapping passage.

Computed variables

These are calculated during the statistical transformation and can be accessed with delayed evaluation.

`axis` unique axis identifier (integer)

`lower,upper` distances to endpoints from origin (before offset)

`yintercept,xintercept` intercepts (possibly `Inf`) of offset axis

References

Gower JC, Gardner–Lubbe S, & le Roux NJ (2011) *Understanding Biplots*. Wiley, ISBN: 978-0-470-01255-0. https://www.wiley.com/go/biplots

Gower JC & Harding SA (1988) "Nonlinear biplots". *Biometrika* **75**(3): 445–455. doi:10.1093/biomet/75.3.445

See Also

Other stat layers: `stat_bagplot()`, `stat_center()`, `stat_chull()`, `stat_cone()`, `stat_depth()`, `stat_scale()`, `stat_spantree()`

Examples

```
# stack loss gradient
stackloss %>%
  lm(formula = stack.loss ~ Air.Flow + Water.Temp + Acid.Conc.) %>%
  coef() %>%
  as.list() %>% as.data.frame() %>%
  subset(select = c(Air.Flow, Water.Temp, Acid.Conc.)) ->
  coef_data
# gradient rule with respect to two predictors
stackloss_centered <- scale(stackloss, scale = FALSE)
stackloss_centered %>%
  ggplot(aes(x = Acid.Conc., y = Air.Flow)) +
  coord_square() +
  geom_point(aes(size = stack.loss, alpha = sign(stack.loss))) +
  scale_size_area() + scale_alpha_binned(breaks = c(-1, 0, 1)) +
  stat_rule(
    geom = "axis",
    data = coef_data,
    referent = stackloss_centered,
    fun.offset = function(x) minabspp(x, p = .5)
  )
```

stat_scale *Multiply artificial coordinates by a scale factor*

Description

This is a simple stat that applies a constant scale factor to both positional coordinates. It can be handy in tandem with secondary axes.

Usage

```
stat_scale(
  mapping = NULL,
  data = NULL,
  geom = "point",
  position = "identity",
  show.legend = NA,
  inherit.aes = TRUE,
  ...,
  mult = 1
)
```

Arguments

mapping	Set of aesthetic mappings created by aes(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
geom	The geometric object to use to display the data for this layer. When using a stat_*() function to construct a layer, the geom argument can be used to override the default coupling between stats and geoms. The geom argument accepts the following: • A Geom ggproto subclass, for example GeomPoint. • A string naming the geom. To give the geom as a string, strip the function name of the geom_ prefix. For example, to use geom_point(), give the geom as "point". • For more information and other ways to specify the geom, see the layer geom documentation.

position — A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following:

- The result of calling a position function, such as position_jitter(). This method allows for passing extra arguments to the position.
- A string naming the position adjustment. To give the position as a string, strip the function name of the position_ prefix. For example, to use position_jitter(), give the position as "jitter".
- For more information and other ways to specify the position, see the layer position documentation.

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().

... — Additional arguments passed to ggplot2::layer().

mult — Numeric value used to scale the coordinates.

Value

A ggproto layer.

Multidimensional position aesthetics

This statistical transformation is compatible with the convenience function aes_coord().

Some transformations (e.g. stat_center()) commute with projection to the lower (1 or 2)-dimensional biplot space. If they detect aesthetics of the form ..coord[0-9]+, then ..coord1 and ..coord2 are converted to x and y while any remaining are ignored.

Other transformations (e.g. stat_spantree()) yield different results in a lower-dimensional biplot when they are computed before versus after projection. If the stat layer detects these aesthetics, then the transformation is performed before projection, and the results in the first two dimensions are returned as x and y.

A small number of transformations (stat_rule()) are incompatible with these aesthetics but will accept aes_coord() without warning.

See Also

Other stat layers: stat_bagplot(), stat_center(), stat_chull(), stat_cone(), stat_depth(), stat_rule(), stat_spantree()

stat_spantree *Calculate a minimum spanning tree among cases or variables*

Description

This stat layer identifies the $n - 1$ pairs among n points that form a minimum spanning tree, then calculates the segments between these poirs in the two dimensions `x` and `y`.

Usage

```
stat_spantree(
  mapping = NULL,
  data = NULL,
  geom = "segment",
  position = "identity",
  engine = "mlpack",
  method = "euclidean",
  show.legend = NA,
  inherit.aes = TRUE,
  ...
)
```

Arguments

`mapping`	Set of aesthetic mappings created by `aes()`. If specified and `inherit.aes = TRUE` (the default), it is combined with the default mapping at the top level of the plot. You must supply `mapping` if there is no plot mapping.
`data`	The data to be displayed in this layer. There are three options: If `NULL`, the default, the data is inherited from the plot data as specified in the call to `ggplot()`. A `data.frame`, or other object, will override the plot data. All objects will be fortified to produce a data frame. See `fortify()` for which variables will be created. A `function` will be called with a single argument, the plot data. The return value must be a `data.frame`, and will be used as the layer data. A `function` can be created from a `formula` (e.g. `~ head(.x, 10)`).
`geom`	The geometric object to use to display the data for this layer. When using a `stat_*()` function to construct a layer, the `geom` argument can be used to override the default coupling between stats and geoms. The `geom` argument accepts the following: • A `Geom` ggproto subclass, for example `GeomPoint`. • A string naming the geom. To give the geom as a string, strip the function name of the `geom_` prefix. For example, to use `geom_point()`, give the geom as `"point"`. • For more information and other ways to specify the geom, see the layer geom documentation.

position	A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following: • The result of calling a position function, such as position_jitter(). This method allows for passing extra arguments to the position. • A string naming the position adjustment. To give the position as a string, strip the function name of the position_ prefix. For example, to use position_jitter(), give the position as "jitter". • For more information and other ways to specify the position, see the layer position documentation.
engine	A single character string specifying the package implementation to use; "mlpack", "vegan", or "ade4".
method	Passed to stats::dist() if engine is "vegan" or "ade4", ignored if "mlpack".
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().
...	Additional arguments passed to ggplot2::layer().

Details

A minimum spanning tree (MST) on the point cloud X is a minimal connected graph on X with the smallest possible sum of distances (or dissimilarities) between linked points. These layers call stats::dist() to calculate a distance/dissimilarity object and an engine from **mlpack**, **vegan**, or **ade4** to calculate the MST. The result is formatted with position aesthetics readable by ggplot2::geom_segment().

An MST calculated on x and y reflects the distances among the points in X in the reduced-dimension plane of the biplot. In contrast, one calculated on the full set of coordinates reflects distances in higher-dimensional space. Plotting this high-dimensional MST on the 2-dimensional biplot provides a visual cue as to how faithfully two dimensions can encapsulate the "true" distances between points (Jolliffe, 2002).

Value

A ggproto layer.

Multidimensional position aesthetics

This statistical transformation is compatible with the convenience function aes_coord().

Some transformations (e.g. stat_center()) commute with projection to the lower (1 or 2)-dimensional biplot space. If they detect aesthetics of the form ..coord[0-9]+, then ..coord1 and ..coord2 are converted to x and y while any remaining are ignored.

Other transformations (e.g. stat_spantree()) yield different results in a lower-dimensional biplot when they are computed before versus after projection. If the stat layer detects these aesthetics,

then the transformation is performed before projection, and the results in the first two dimensions are returned as `x` and `y`.

A small number of transformations (`stat_rule()`) are incompatible with these aesthetics but will accept `aes_coord()` without warning.

Computed variables

These are calculated during the statistical transformation and can be accessed with delayed evaluation.

`xend,yend,x,y` endpoints of tree branches (segments)

References

Jolliffe IT (2002) *Principal Component Analysis*, Second Edition. Springer Series in Statistics, ISSN 0172-7397. doi:10.1007/b98835 https://link.springer.com/book/10.1007/b98835

See Also

Other stat layers: `stat_bagplot()`, `stat_center()`, `stat_chull()`, `stat_cone()`, `stat_depth()`, `stat_rule()`, `stat_scale()`

Examples

```
eurodist %>%
  cmdscale(k = 6) %>%
  as.data.frame() %>%
  tibble::rownames_to_column(var = "city") ->
  euro_mds
ggplot(euro_mds, aes(V1, V2, label = city)) +
  stat_spantree() +
  geom_label(alpha = .25)
ggplot(euro_mds, aes_c(aes_coord(euro_mds, "V"), aes(label = city))) +
  stat_spantree() +
  geom_label(aes(x = V1, y = V2), alpha = .25)
```

Index