Package 'ggmulti'

January 5, 2021

Type Package

Title High Dimensional Data Visualization

Version 0.1.0

Description It provides materials (i.e. 'serial axes' objects, Andrew's plot, various glyphs for scatter plot) to visualize high dimensional data.

License GPL-2

Depends R (>= 3.4.0), methods, ggplot2

Imports stats, utils, grid, dplyr, tidyr

Suggests png, tools, stringr, magrittr, gridExtra, tibble, testthat, grDevices, knitr, rmarkdown, tidyverse, gtable, covr, maps, nycflights13, ggplot2movies

LazyData true

RoxygenNote 7.1.1

Encoding UTF-8

VignetteBuilder knitr

Language en-US

NeedsCompilation no

Author Zehao Xu [aut, cre], R. Wayne Oldford [aut]

Maintainer Zehao Xu <z267xu@uwaterloo.ca>

Repository CRAN

Date/Publication 2021-01-05 10:00:02 UTC

R topics documented:

`add_serialaxes_layers` *Layers for serial axes coordinate*

Description

Project the regular `geom` layers onto the serial axes coordinate.

Usage

```
add_serialaxes_layers(layer, plot, object, axes)
```

Arguments

`layer`	a layer object
`plot`	a `ggplot` object
`object`	some parameters used to modify this serial axes `ggplot` object (i.e. `axes.sequence`, ...)
`axes`	canvas sequence axes

Details

The class is determined by layers you add. For example, you want to add a boxplot layer on serial axes coordinate. By the ggplot syntax, it should be `ggplot(data,mapping) + geom_boxplot() + coord_serialaxes()` To make it work, object `add_serialaxes_layers.GeomBoxplot` must be created. In this function, some computations will be applied.

coord_polar *Polar coordinates*

Description

The polar coordinate system is most commonly used for pie charts, which are a stacked bar chart in polar coordinates.

Usage

```
coord_polar(
  theta = "x",
  start = 0,
  direction = 1,
  clip = "on",
  is_linear = FALSE
)

coord_polar(
  theta = "x",
  start = 0,
  direction = 1,
  clip = "on",
  is_linear = FALSE
)
```

Arguments

theta	variable to map angle to (x or y)
start	Offset of starting point from 12 o'clock in radians. Offset is applied clockwise or anticlockwise depending on value of direction.
direction	1, clockwise; -1, anticlockwise
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. For details, please see coord_cartesian().
is_linear	Returns TRUE if the coordinate system is linear; FALSE otherwise. Mainly used for radial axes

Examples

```
# NOTE: Use these plots with caution - polar coordinates has
# major perceptual problems.  The main point of these examples is
# to demonstrate how these common plots can be described in the
# grammar.  Use with EXTREME caution.

#' # A pie chart = stacked bar chart + polar coordinates
```

```
pie <- ggplot(mtcars, aes(x = factor(1), fill = factor(cyl))) +
 geom_bar(width = 1)
pie + coord_polar(theta = "y")



# A coxcomb plot = bar chart + polar coordinates
cxc <- ggplot(mtcars, aes(x = factor(cyl))) +
  geom_bar(width = 1, colour = "black")
cxc + coord_polar()
# A new type of plot?
cxc + coord_polar(theta = "y")

# The bullseye chart
pie + coord_polar()

# Hadley's favourite pie chart
df <- data.frame(
  variable = c("does not resemble", "resembles"),
  value = c(20, 80)
)
ggplot(df, aes(x = "", y = value, fill = variable)) +
  geom_col(width = 1) +
  scale_fill_manual(values = c("red", "yellow")) +
  coord_polar("y", start = pi / 3) +
  labs(title = "Pac man")

# Windrose + doughnut plot
if (require("ggplot2movies")) {
movies$rrating <- cut_interval(movies$rating, length = 1)
movies$budgetq <- cut_number(movies$budget, 4)

doh <- ggplot(movies, aes(x = rrating, fill = budgetq))

# Wind rose
doh + geom_bar(width = 1) + coord_polar()
# Race track plot
doh + geom_bar(width = 0.9, position = "fill") + coord_polar(theta = "y")
}
```

`coord_radar` *Radar axes*

Description

A radar (spider) coordinate.

A radar (spider) coordinate. A wrapper of the function `coord_polar()` by forcing it linear.

Usage

```
coord_radar(theta = "x", start = 0, direction = 1, clip = "on")

coord_radar(theta = "x", start = 0, direction = 1, clip = "on")
```

Arguments

theta	variable to map angle to (x or y)
start	Offset of starting point from 12 o'clock in radians. Offset is applied clockwise or anticlockwise depending on value of direction.
direction	1, clockwise; -1, anticlockwise
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. For details, please see coord_cartesian().

Examples

```
ggplot(iris, mapping = aes(colour = Species)) +
  geom_serialaxes(axes.sequence = c(colnames(iris), colnames(iris)[1])) +
  coord_radar()
ggplot(iris, mapping = aes(colour = Species)) +
  geom_serialaxes(axes.sequence = c(colnames(iris), colnames(iris)[1])) +
  coord_radar()
```

---

coord_serialaxes *Serial axes coordinates*

Description

It is mainly used to visualize the high dimensional data set either on the parallel coordinate or the radial coordinate.

Usage

```
coord_serialaxes(
  axes.layout = c("parallel", "radial"),
  scaling = c("variable", "observation", "data", "none"),
  axes.sequence = character(0L),
  positive = TRUE,
  ...
)
```

Arguments

axes.layout	Serial axes layout, either "parallel" or "radial".
scaling	One of 'variable', 'data', 'observation' or 'none' to specify how the data is scaled.
axes.sequence	A vector with variable names that defines the axes sequence.
positive	If 'y' is set as the density estimate, where the smoothed curved is faced to, right ('positive') or left ('negative') as vertical layout; up ('positive') or down ('negative') as horizontal layout?
...	other arguments used to modify layers

Details

Serial axes coordinate system (parallel or radial) is different from the Cartesian coordinate system or its transformed system (say `polar` in `ggplot2`) since it does not have a formal transformation (i.e. in polar coordinate system, "x = rcos(theta)", "y = rsin(theta)"). In serial axes coordinate system, mapping aesthetics does not really require "x" or "y". Any "non-aesthetics" components passed in the 'mapping' system will be treated as an individual axis.

To project a common `geom` layer on such serialaxes, users can customize function `add_serialaxes_layers`.

Examples

```
# set sequence by `axes.sequence`
p <- ggplot(iris) +
       geom_path(alpha = 0.2) +
       coord_serialaxes(axes.sequence = colnames(iris))
# an 'iris' parallel coordinate plot.
p
# histogram layer (parallel coord)
p + geom_histogram(alpha = 0.8, mapping = aes(fill = Species))
# density layer
p + geom_density(alpha = 0.8)
# quantile layer
p + geom_quantiles(alpha = 0.8, colour = "red", size = 2)

# radial axes
# set sequence in `mapping`
ggplot(iris,
       mapping = aes(
         Sepal.Length = Sepal.Length,
         Sepal.Width = Sepal.Width,
         Petal.Length = Petal.Length,
         Petal.Width = Petal.Width,
         colour = Species
       )) +
       geom_path() +
       coord_serialaxes(axes.layout = "radial")
```

dot_product *Transformation Coefficients*

Description

The dimension of the original data set is n*p. It can be projected onto a n*k space. The functions below are to provide such transformations, e.g. the Andrews coefficient (a Fourier transformation) and the Legendre polynomials.

Usage

```
andrews(p = 4, k = 50 * (p - 1), ...)

legendre(p = 4, k = 50 * (p - 1), ...)
```

Arguments

p	The number of dimensions
k	The sequence length
...	Other arguments passed on to methods. Mainly used for customized transformation function

Value

A list contains two named components

1. vector: A length k vector (define the domain)
2. matrix: A p*k transformed coefficient matrix

References

Andrews, David F. "Plots of high-dimensional data." *Biometrics* (1972): 125-136.

Abramowitz, Milton, and Irene A. Stegun, eds. "Chapter 8" *Handbook of mathematical functions with formulas, graphs, and mathematical tables*. Vol. 55. US Government printing office, 1948.

Examples

```
x <- andrews(p = 4)
dat <- iris[, -5]
proj <- t(as.matrix(dat) %*% x$matrix)
matplot(x$vector, proj,
        type = "l", lty = 1,
        col = "black",
        xlab = "x",
        ylab = "Andrews coefficients",
        main = "Iris")
```

Geom-ggproto *Base Geom ggproto classes for ggplot2*

Description

All geom_ functions (like geom_point) return a layer that contains a Geom object (like GeomPoint). The Geom object is responsible for rendering the data in the plot. Each of the Geom objects is a ggproto object, descended from the top-level Geom, and each implements various methods and fields. Compared to Stat and Position, Geom is a little different because the execution of the setup and compute functions is split up. setup_data runs before position adjustments, and draw_layer is not run until render time, much later. This means there is no setup_params because it's hard to communicate the changes.

Usage

```
GeomDensity_

GeomBar_

GeomQuantiles

GeomSerialaxesDensity

GeomSerialaxesHist

GeomSerialaxesQuantile

GeomSerialaxes
```

Format

An object of class GeomDensity_ (inherits from GeomRibbon, Geom, ggproto, gg) of length 6.

An object of class GeomBar_ (inherits from GeomBar, GeomRect, Geom, ggproto, gg) of length 4.

An object of class GeomQuantiles (inherits from GeomQuantile, GeomPath, Geom, ggproto, gg) of length 1.

An object of class GeomSerialaxesDensity (inherits from GeomDensity_, GeomRibbon, Geom, ggproto, gg) of length 2.

An object of class GeomSerialaxesHist (inherits from GeomBar_, GeomBar, GeomRect, Geom, ggproto, gg) of length 2.

An object of class GeomSerialaxesQuantile (inherits from GeomPath, Geom, ggproto, gg) of length 4.

An object of class GeomSerialaxes (inherits from GeomPath, Geom, ggproto, gg) of length 2.

geom_density_ *More general smoothed density estimates*

Description

Computes and draws kernel density estimate. Compared with `geom_density()`, it provides more general cases that accepting ‘x‘ and ‘y‘. The ‘x‘ (or ‘y‘) is a group variable and ‘y‘ (or ‘x‘) is the target variable to be plotted. The result is a different density of ‘y‘ (‘x‘) for each value of ‘x‘ (‘y‘). If only one of ‘x‘ or ‘y‘ is provided, it will be the target variable (no grouping) and the standard `geom_density()` will be executed.

Usage

```
geom_density_(
  mapping = NULL,
  data = NULL,
  stat = "density_",
  position = "identity_",
  ...,
  scale.x = NULL,
  scale.y = c("data", "variable"),
  as.mix = FALSE,
  positive = TRUE,
  adjust = 0.9,
  na.rm = FALSE,
  orientation = NA,
  show.legend = NA,
  inherit.aes = TRUE
)

stat_density_(
  mapping = NULL,
  data = NULL,
  geom = "density_",
  position = "stack_",
  ...,
  bw = "nrd0",
  adjust = 1,
  kernel = "gaussian",
  n = 512,
  trim = FALSE,
  na.rm = FALSE,
  orientation = NA,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by aes() or aes_(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x,10)).
stat	Use to override the default connection between geom_density and stat_density.
position	Position adjustment, either as a string, or the result of a call to a position adjustment function.
...	Other arguments passed on to layer(). These are often aesthetics, used to set an aesthetic to a fixed value, like colour = "red" or size = 3. They may also be parameters to the paired geom/stat.
scale.x	A length 2 numerical vector. Scale the n coordinates of the points where the density is estimated.
scale.y	one of 'data', 'variable' to specify.

Type	Description
data (default)	The density estimates are scaled by the whole data set
variable	The density estimates are scaled by each variable

If the scale.y is "data", it is meaningful to compare the density (shape and area) across all groups; else it is only meaningful to compare the density under each variable.

as.mix	Logical. Under each variable, if as.mix = TRUE, the sum of the density estimate area is mixed and scaled to maximum 1. The area of each group is proportional to its own count; if as.mix = FALSE the area of each group is the same, with maximum 1.
positive	If 'y' is set as the density estimate, where the smoothed curved is faced to, right ('positive') or left ('negative') as vertical layout; up ('positive') or down ('negative') as horizontal layout?
adjust	adjust the proportional maximum height of the estimate (density, histogram, ...).
na.rm	If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.
orientation	The orientation of the layer. The default (NA) automatically determines the orientation from the aesthetic mapping. In the rare event that this fails it can be given explicitly by setting orientation to either "x" or "y". See the *Orientation* section for more detail.

show.legend	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.
inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. borders().
geom	Use to override the default connection between geom_density and stat_density.
bw	The smoothing bandwidth to be used. If numeric, the standard deviation of the smoothing kernel. If character, a rule to choose the bandwidth, as listed in stats::bw.nrd().
kernel	Kernel. See list of available kernels in density().
n	number of equally spaced points at which the density is to be estimated, should be a power of two, see density() for details
trim	If FALSE, the default, each density is computed on the full range of the data. If TRUE, each density is computed over the range of that group: this typically means the estimated x values will not line-up, and hence you won't be able to stack density values. This parameter only matters if you are displaying multiple densities in one plot or if you are manually adjusting the scale limits.

Details

There are four combinations of scale.y and as.mix

scale.y = **"variable" and** as.mix = **FALSE** The density estimates area of each group under the same variable is the same and scaled to maximum of 1.

scale.y = **"variable" and** as.mix = **TRUE** The density estimates area of each group under the same variable is proportional to its own counts (over this variable).

scale.y = **"data" and** as.mix = **FALSE** The sum of density estimates area of all group is scaled to maximum of 1. The sum of the density area for each variable is proportional to the its counts (over the whole dataset). Under each variable, the area of each group is the same.

scale.y = **"data" and** as.mix = **TRUE** The sum of density estimates area of all group is scaled to maximum of 1 and the area of each group is proportional to its own count.

Orientation

This geom treats each axis differently and, thus, can thus have two orientations. Often the orientation is easy to deduce from a combination of the given mappings and the types of positional scales in use. Thus, ggplot2 will by default try to guess which orientation the layer should have. Under rare circumstances, the orientation is ambiguous and guessing may fail. In that case the orientation can be specified directly using the orientation parameter, which can be either "x" or "y". The value gives the axis that the geom should run along, "x" being the default orientation you would expect for the geom.

See Also

geom_density, geom_hist_

Examples

```
if(require(dplyr)) {
  mpg %>%
    dplyr::filter(drv != "f") %>%
    ggplot(mapping = aes(x = drv, y = cty, fill = factor(cyl))) +
    geom_density_(alpha = 0.1)

  # only `x` or `y` is provided
  # that would be equivalent to call function `geom_density()`
  diamonds %>%
    dplyr::sample_n(500) %>%
    ggplot(mapping = aes(x = price)) +
    geom_density_()

  # density and boxplot
  # set the density estimate on the left
  mpg %>%
    dplyr::filter(drv != "f") %>%
    ggplot(mapping = aes(x = drv, y = cty, fill = factor(cyl))) +
    geom_density_(alpha = 0.1, scale.y = "data", positive = FALSE) +
    geom_boxplot()

  # x as density
  set.seed(12345)
  suppressWarnings(
    diamonds %>%
      dplyr::sample_n(500) %>%
      ggplot(mapping = aes(x = price, y = cut, fill = color)) +
      geom_density_(orientation = "x", adjust = 0.25,
                    position = "stack_",
                    scale.y = "variable")
  )
}
# settings of `scale.y` and `as.mix`

ggplots <- lapply(list(
                      list(scale.y = "data", as.mix = TRUE),
                      list(scale.y = "data", as.mix = FALSE),
                      list(scale.y = "variable", as.mix = TRUE),
                      list(scale.y = "variable", as.mix = FALSE)
                     ),
                    function(vars) {
                      scale.y <- vars[["scale.y"]]
                      as.mix <- vars[["as.mix"]]
                      ggplot(mpg,
                             mapping = aes(x = drv, y = cty, fill = factor(cyl))) +
                        geom_density_(alpha = 0.1, scale.y = scale.y, as.mix = as.mix) +
                        labs(title = paste("scale.y =", scale.y),
                             subtitle = paste("as.mix =", as.mix))
                    })
suppressWarnings(
  gridExtra::grid.arrange(grobs = ggplots)
```

```
)
```

`geom_hist_` *More general histogram*

Description

More general histogram (`geom_histogram()`) or bar plot (`geom_bar()`). Both ‘x‘ and ‘y‘ could be accommodated. ‘x‘ (or ‘y‘) is a group variable and ‘y‘ (or ‘x‘) the target variable to be plotted. The result is a different histogram of ‘y‘ (‘x‘) for each value of ‘x‘ (‘y‘). If only one of ‘x‘ or ‘y‘ is provided, it will be the target variable (no grouping) and the standard `geom_histogram()` will be executed.

Usage

```
geom_hist_(
  mapping = NULL,
  data = NULL,
  stat = "hist_",
  position = "stack_",
  ...,
  scale.x = NULL,
  scale.y = c("data", "variable"),
  as.mix = FALSE,
  binwidth = NULL,
  bins = NULL,
  positive = TRUE,
  adjust = 0.9,
  na.rm = FALSE,
  orientation = NA,
  show.legend = NA,
  inherit.aes = TRUE
)

geom_histogram_(
  mapping = NULL,
  data = NULL,
  stat = "bin_",
  position = "stack_",
  ...,
  scale.x = NULL,
  scale.y = c("data", "variable"),
  as.mix = FALSE,
  positive = TRUE,
  adjust = 0.9,
  binwidth = NULL,
```

```
  bins = NULL,
  na.rm = FALSE,
  orientation = NA,
  show.legend = NA,
  inherit.aes = TRUE
)

geom_bar_(
  mapping = NULL,
  data = NULL,
  stat = "count_",
  position = "stack_",
  ...,
  scale.x = NULL,
  scale.y = c("data", "variable"),
  positive = TRUE,
  adjust = 0.9,
  na.rm = FALSE,
  orientation = NA,
  show.legend = NA,
  inherit.aes = TRUE
)

stat_hist_(
  mapping = NULL,
  data = NULL,
  geom = "bar_",
  position = "stack_",
  ...,
  binwidth = NULL,
  bins = NULL,
  center = NULL,
  boundary = NULL,
  breaks = NULL,
  closed = c("right", "left"),
  pad = FALSE,
  width = NULL,
  na.rm = FALSE,
  orientation = NA,
  show.legend = NA,
  inherit.aes = TRUE
)

stat_bin_(
  mapping = NULL,
  data = NULL,
  geom = "bar_",
  position = "stack_",
```

```
  ...,
  binwidth = NULL,
  bins = NULL,
  center = NULL,
  boundary = NULL,
  breaks = NULL,
  closed = c("right", "left"),
  pad = FALSE,
  na.rm = FALSE,
  orientation = NA,
  show.legend = NA,
  inherit.aes = TRUE
)

stat_count_(
  mapping = NULL,
  data = NULL,
  geom = "bar_",
  position = "stack_",
  ...,
  width = NULL,
  na.rm = FALSE,
  orientation = NA,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by aes() or aes_(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x,10)).
position	Position adjustment, either as a string, or the result of a call to a position adjustment function. Function 'geom_hist_' and 'geom_histogram_' understand 'stack_' (stacks bars on top of each other), or 'dodge_' () and 'dodge2_' (overlapping objects side-to-side) instead of 'stack', 'dodge' or 'dodge2'
...	Other arguments passed on to layer(). These are often aesthetics, used to set an aesthetic to a fixed value, like colour = "red" or size = 3. They may also be parameters to the paired geom/stat.

`scale.x` A length 2 numerical vector. Scale the n coordinates of the points where the density is estimated.

`scale.y` one of 'data', 'variable' to specify.

Type	Description
data (default)	The density estimates are scaled by the whole data set
variable	The density estimates are scaled by each variable

If the `scale.y` is "data", it is meaningful to compare the density (shape and area) across all groups; else it is only meaningful to compare the density under each variable.

`as.mix` Logical. Under each variable, if `as.mix = TRUE`, the sum of the density estimate area is mixed and scaled to maximum 1. The area of each group is proportional to its own count; if `as.mix = FALSE` the area of each group is the same, with maximum 1.

`binwidth` The width of the bins. Can be specified as a numeric value or as a function that calculates width from unscaled x. Here, "unscaled x" refers to the original x values in the data, before application of any scale transformation. When specifying a function along with a grouping structure, the function will be called once per group. The default is to use the number of bins in `bins`, covering the range of the data. You should always override this value, exploring multiple widths to find the best to illustrate the stories in your data.

The bin width of a date variable is the number of days in each time; the bin width of a time variable is the number of seconds.

`bins` Number of bins. Overridden by `binwidth`. Defaults to 30.

`positive` If 'y' is set as the density estimate, where the smoothed curved is faced to, right ('positive') or left ('negative') as vertical layout; up ('positive') or down ('negative') as horizontal layout?

`adjust` adjust the proportional maximum height of the estimate (density, histogram, ...).

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

`orientation` The orientation of the layer. The default (`NA`) automatically determines the orientation from the aesthetic mapping. In the rare event that this fails it can be given explicitly by setting `orientation` to either `"x"` or `"y"`. See the *Orientation* section for more detail.

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

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

`geom, stat` Use to override the default connection between geom_hist_()/geom_histogram_()/geom_bar_() and stat_hist_()/stat_bin_()/stat_count_().

`center` bin position specifiers. Only one, `center` or `boundary`, may be specified for a single plot. `center` specifies the center of one of the bins. `boundary` specifies

the boundary between two bins. Note that if either is above or below the range of the data, things will be shifted by the appropriate integer multiple of `binwidth`. For example, to center on integers use `binwidth = 1` and `center = 0`, even if `0` is outside the range of the data. Alternatively, this same alignment can be specified with `binwidth = 1` and `boundary = 0.5`, even if `0.5` is outside the range of the data.

`boundary` bin position specifiers. Only one, `center` or `boundary`, may be specified for a single plot. `center` specifies the center of one of the bins. `boundary` specifies the boundary between two bins. Note that if either is above or below the range of the data, things will be shifted by the appropriate integer multiple of `binwidth`. For example, to center on integers use `binwidth = 1` and `center = 0`, even if `0` is outside the range of the data. Alternatively, this same alignment can be specified with `binwidth = 1` and `boundary = 0.5`, even if `0.5` is outside the range of the data.

`breaks` Alternatively, you can supply a numeric vector giving the bin boundaries. Overrides `binwidth`, `bins`, `center`, and `boundary`.

`closed` One of `"right"` or `"left"` indicating whether right or left edges of bins are included in the bin.

`pad` If `TRUE`, adds empty bins at either end of x. This ensures frequency polygons touch 0. Defaults to `FALSE`.

`width` Bar width. By default, set to 90% of the resolution of the data.

Details

1. 'geom_hist_' is a wrapper of 'geom_histogram_' and 'geom_count_'. In other words, suppose the 'y' is our interest, `geom_hist_()` can accommodate both continuous or discrete "y" but `geom_histogram_()` is only for the continuous "y" and `geom_bar_()` is only for the discrete "y".

2. There are four combinations of `scale.y` and `as.mix`

`scale.y` **= "variable" and** `as.mix` **= FALSE** The density estimates area of each group under the same variable is the same and scaled to maximum of 1.

`scale.y` **= "variable" and** `as.mix` **= TRUE** The density estimates area of each group under the same variable is proportional to its own counts (over this variable).

`scale.y` **= "data" and** `as.mix` **= FALSE** The sum of density estimates area of all group is scaled to maximum of 1. The sum of the density area for each variable is proportional to the its counts (over the whole dataset). Under each variable, the area of each group is the same.

`scale.y` **= "data" and** `as.mix` **= TRUE** The sum of density estimates area of all group is scaled to maximum of 1 and the area of each group is proportional to its own count.

Note that, if it is a grouped bar chart (both 'x' and 'y' are categorical), parameter 'as.mix' is meaningless.

Orientation

This geom treats each axis differently and, thus, can thus have two orientations. Often the orientation is easy to deduce from a combination of the given mappings and the types of positional scales in use. Thus, ggplot2 will by default try to guess which orientation the layer should have. Under

rare circumstances, the orientation is ambiguous and guessing may fail. In that case the orientation can be specified directly using the `orientation` parameter, which can be either `"x"` or `"y"`. The value gives the axis that the geom should run along, `"x"` being the default orientation you would expect for the geom.

See Also

geom_histogram, geom_density_

Examples

```
if(require(dplyr) && require(tidyr)) {

  # histogram
  p0 <- mpg %>%
    dplyr::filter(manufacturer %in% c("dodge", "ford", "toyota", "volkswagen")) %>%
    ggplot(mapping = aes(x = manufacturer, y = cty))
  p0 + geom_hist_()

  ## set position
  #### default is "stack_"
  p0 + geom_hist_(mapping = aes(fill = fl))
  #### "dodge_"
  p0 + geom_hist_(position = "dodge_",
                  mapping = aes(fill = fl))
  #### "dodge2_"
  p0 + geom_hist_(position = "dodge2_",
                  mapping = aes(fill = fl))

  # bar chart
  mpg %>%
    ggplot(mapping = aes(x = drv, y = class)) +
    geom_hist_(orientation = "y")

  # scale.y as "variable"
  p <- iris %>%
    tidyr::pivot_longer(cols = -Species,
                        names_to = "Outer sterile whorls",
                        values_to = "x") %>%
    ggplot(mapping = aes(x = `Outer sterile whorls`,
                         y = x, fill = Species)) +
    stat_hist_(scale.y = "variable",
               adjust = 0.6,
               alpha = 0.5)
  p
  # with density on the left
  p + stat_density_(scale.y = "variable",
                    adjust = 0.6,
                    alpha = 0.5,
                    positive = FALSE)

  ############ only `x` or `y` is provided ############
```

```
  # that would be equivalent to call function
  # `geom_histogram()` or `geom_bar()`
  ### histogram
  diamonds %>%
    dplyr::sample_n(500) %>%
    ggplot(mapping = aes(x = price)) +
    geom_hist_()
  ### bar chart
  diamonds %>%
    dplyr::sample_n(500) %>%
    ggplot(mapping = aes(x = cut)) +
    geom_hist_()
}
```

`geom_image_glyph` *Add image glyphs on scatter plot*

Description

Each point glyph can be an image (png, jpeg, etc) object.

Usage

```
geom_image_glyph(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  ...,
  images,
  imagewidth = 1.2,
  imageheight = 0.9,
  units = "cm",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

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

	A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x,10)).
stat	The statistical transformation to use on the data for this layer, as a string.
position	Position adjustment, either as a string, or the result of a call to a position adjustment function.
...	Other arguments passed on to layer(). These are often aesthetics, used to set an aesthetic to a fixed value, like colour = "red" or size = 3. They may also be parameters to the paired geom/stat.
images	a list of images (a raster object, bitmap image). If not provided, geom_point() will be executed.
imagewidth	Numerical; width of image
imageheight	Numerical; height of image
units	A character vector specifying the units for the image height and width, see unit; default is "cm" (force the width and height).
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 geom layer

Aesthetics

geom_..._glyph() understands the following aesthetics (required aesthetics are in bold):

- **x**
- **y**
- alpha
- colour
- fill
- group
- shape
- size
- stroke
- linetype

See Also

geom_serialaxes_glyph, geom_polygon_glyph

Examples

```
# image glyph
if(requireNamespace("png")) {

# Download images from github
img_path <- list.files(file.path(find.package(package = 'ggmulti'),
                                 "images"),
                       full.names = TRUE)
Raptors <- png::readPNG(img_path[1L])

p <- ggplot(data = data.frame(x = 0, y = 0),
            mapping = aes(x = x, y = y)) +
       geom_image_glyph(images = Raptors,
                        units = "native",
                        imagewidth = 1,
                        imageheight = 1)
p

}
```

---

geom_polygon_glyph *Add polygon glyphs on scatter plot*

Description

Each point glyph can be a polygon object. We provide some common polygon coords in polygon_glyph. Also, users can customize their own polygons.

Usage

```
geom_polygon_glyph(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  ...,
  polygon_x,
  polygon_y,
  linewidth = 1,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by aes() or aes_(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x,10)).
stat	The statistical transformation to use on the data for this layer, as a string.
position	Position adjustment, either as a string, or the result of a call to a position adjustment function.
...	Other arguments passed on to layer(). These are often aesthetics, used to set an aesthetic to a fixed value, like colour = "red" or size = 3. They may also be parameters to the paired geom/stat.
polygon_x	nested list of x-coordinates of polygons, one list element for each scatterplot point. If not provided, geom_point() will be executed.
polygon_y	nested list of y-coordinates of polygons, one list element for each scatterplot point. If not provided, geom_point() will be executed.
linewidth	line width of the "glyph" object
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 geom layer

Aesthetics

geom_..._glyph() understands the following aesthetics (required aesthetics are in bold):

- **x**
- **y**
- alpha

- colour
- fill
- group
- shape
- size
- stroke
- linetype

See Also

geom_serialaxes_glyph, geom_image_glyph

Examples

```
# polygon glyph
p <- ggplot(data = data.frame(x = 1:4, y = 1:4),
            mapping = aes(x = x, y = y)) +
  geom_polygon_glyph(polygon_x = list(x_star, x_cross, x_hexagon, x_airplane),
                     polygon_y = list(y_star, y_cross, y_hexagon, y_airplane),
                     colour = 'black', fill = 'red')
p

# the coords of each polygons can be achieved by calling function `ggplot_build`
build <- ggplot2::ggplot_build(p)
polygon_x <- build$data[[1]]$polygon_x
polygon_y <- build$data[[1]]$polygon_y
```

---

geom_quantiles *Add quantile layers on serial axes coordinate*

Description

In ggplot2, geom_quantile() is used to fit a quantile regression to the data and draws the fitted quantiles with lines. However, geom_quantiles() is mainly used to draw quantile lines on serial axes. See examples

Usage

```
geom_quantiles(
  mapping = NULL,
  data = NULL,
  stat = "quantile",
  position = "identity",
  ...,
  lineend = "butt",
  linejoin = "round",
```

```
  linemitre = 10,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by aes() or aes_(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x,10)).
stat	Use to override the default connection between geom_quantile and stat_quantile.
position	Position adjustment, either as a string, or the result of a call to a position adjustment function.
...	Other arguments passed on to layer(). These are often aesthetics, used to set an aesthetic to a fixed value, like colour = "red" or size = 3. They may also be parameters to the paired geom/stat.
lineend	Line end style (round, butt, square).
linejoin	Line join style (round, mitre, bevel).
linemitre	Line mitre limit (number greater than 1).
na.rm	If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.
show.legend	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.
inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. borders().

See Also

geom_serialaxes_quantile

Examples

```
p <- ggplot(iris,
            mapping = aes(
              Sepal.Length = Sepal.Length,
              Sepal.Width = Sepal.Width,
              Petal.Length = Petal.Length,
              Petal.Width = Petal.Width
            )
  ) +
  geom_path(alpha = 0.2)  +
  coord_serialaxes(scaling = "none")
p + geom_quantiles(colour = c("red", "green", "blue"),
                   quantiles = c(0.25, 0.5, 0.75),
                   size = 2)
```

`geom_serialaxes` *Serial axes layer*

Description

Draw a serial axes layer, parallel axes under Cartesian system and radial axes under Polar system. It only takes the "widens" data. Each non-aesthetics component defined in the mapping `aes()` will be treated as an axis.

Usage

```
geom_serialaxes(
  mapping = NULL,
  data = NULL,
  stat = "serialaxes",
  position = "identity",
  ...,
  axes.sequence = character(0L),
  merge = TRUE,
  na.rm = FALSE,
  orientation = NA,
  show.legend = NA,
  inherit.aes = TRUE
)

stat_serialaxes(
  mapping = NULL,
  data = NULL,
  geom = "serialaxes",
  position = "identity",
  ...,
  axes.sequence = character(0L),
```

```
  merge = TRUE,
  axes.position = NULL,
  scaling = c("variable", "observation", "data", "none"),
  na.rm = FALSE,
  orientation = NA,
  show.legend = NA,
  inherit.aes = TRUE
)

stat_dotProduct(
  mapping = NULL,
  data = NULL,
  geom = "path",
  position = "identity",
  ...,
  axes.sequence = character(0L),
  merge = TRUE,
  scaling = c("variable", "observation", "data", "none"),
  transform = andrews,
  na.rm = FALSE,
  orientation = NA,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

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

	The only difference is that the `mapping` aesthetics will omit the duplicated axes (check examples in `geom_serialaxes`).
`merge`	Should `axes.sequence` be merged with mapping aesthetics as a single mapping `uneval` object?
`na.rm`	If `FALSE`, the default, missing values are removed with a warning. If `TRUE`, missing values are silently removed.
`orientation`	The orientation of the layer. The default (`NA`) automatically determines the orientation from the aesthetic mapping. In the rare event that this fails it can be given explicitly by setting `orientation` to either `"x"` or `"y"`. See the *Orientation* section for more detail.
`show.legend`	logical. Should this layer be included in the legends? `NA`, the default, includes if any aesthetics are mapped. `FALSE` never includes, and `TRUE` always includes. It can also be a named logical vector to finely select the aesthetics to display.
`inherit.aes`	If `FALSE`, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. `borders()`.
`geom`	The geometric object to use display the data
`axes.position`	A numerical vector to determine the axes sequence position; the length should be the same with the length of `axes.sequence` (or mapping `aesthetics`, see examples).
`scaling`	one of 'variable', 'data', 'observation' or 'none' to specify how the data is scaled.
`transform`	A transformation function, can be either 'andrews', 'legendre' or some other customized transformation functions.

Details

The difference between the "lengthens" data and "widens" data can be found in Tidy Data. How to transform one to the other is explained in tidyr

See Also

`coord_radar`, `geom_serialaxes_density`, `geom_serialaxes_quantile`, `geom_serialaxes_hist`

Andrews plot `andrews`, Legendre polynomials `legendre`

Examples

```
# parallel coordinate
p <- ggplot(iris, mapping = aes(Sepal.Length = Sepal.Length,
                                Sepal.Width = Sepal.Width,
                                Petal.Length = Petal.Length,
                                Petal.Width = Petal.Width,
                                colour = Species))
p + geom_serialaxes()

# radial coordinate
p +
  geom_serialaxes() +
```

```
  coord_polar(is_linear = TRUE) # or just call `coord_radar()`!

# andrews plot
p + geom_serialaxes(stat = "dotProduct",
                    transform = andrews, # default
                    scaling = "none")

# Legendre polynomials
p + geom_serialaxes(stat = "dotProduct",
                    transform = legendre,
                    scaling = "none")

# or set the `axes.sequence` rather tahn the mapping `aes`
p <- ggplot(iris) +
       geom_serialaxes(axes.sequence = colnames(iris))
p


############## Determine axes sequence
# 1. set the duplicated axes by mapping aesthetics
ggplot(iris, mapping = aes(Sepal.Length = Sepal.Length,
                           Sepal.Width = Sepal.Width,
                           Sepal.Length = Sepal.Length,
                           Sepal.Width = Sepal.Width,
                           colour = Species)) +
  # only two axes, duplicated axes are removed
  geom_serialaxes()

# 2. set the duplicated axes by axes.sequence
ggplot(iris, mapping = aes(colour = Species)) +
  geom_serialaxes(
    axes.sequence = c("Sepal.Length", "Sepal.Width",
                      "Sepal.Length", "Sepal.Width"))
```

`geom_serialaxes_density` *Smoothed density estimates for "widens" data under serial axes coordinate*

Description

Computes and draws kernel density estimates on serial axes coordinate for each non-aesthetics component defined in the mapping `aes()`.

Usage

```
geom_serialaxes_density(
  mapping = NULL,
  data = NULL,
```

```
  stat = "serialaxes_density",
  position = "identity_",
  ...,
  axes.sequence = character(0L),
  merge = TRUE,
  scale.y = c("data", "variable"),
  as.mix = TRUE,
  positive = TRUE,
  adjust = 0.9,
  na.rm = FALSE,
  orientation = NA,
  show.legend = NA,
  inherit.aes = TRUE
)

stat_serialaxes_density(
  mapping = NULL,
  data = NULL,
  geom = "serialaxes_density",
  position = "stack_",
  ...,
  axes.sequence = character(0L),
  merge = TRUE,
  axes.position = NULL,
  scaling = c("variable", "observation", "data", "none"),
  bw = "nrd0",
  adjust = 1,
  kernel = "gaussian",
  n = 512,
  trim = FALSE,
  na.rm = FALSE,
  orientation = NA,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

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

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

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

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

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

axes.sequence — A vector to define the axes sequence. In serial axes coordinate, the sequence can be either determined in mapping (function aes()) or by axes.sequence. The only difference is that the mapping aesthetics will omit the duplicated axes (check examples in geom_serialaxes).

merge — Should axes.sequence be merged with mapping aesthetics as a single mapping uneval object?

scale.y — one of 'data', 'variable' to specify.

Type	Description
data (default)	The density estimates are scaled by the whole data set
variable	The density estimates are scaled by each variable

If the scale.y is "data", it is meaningful to compare the density (shape and area) across all groups; else it is only meaningful to compare the density under each variable.

as.mix — Logical. Under each variable, if as.mix = TRUE, the sum of the density estimate area is mixed and scaled to maximum 1. The area of each group is proportional to its own count; if as.mix = FALSE the area of each group is the same, with maximum 1.

positive — If 'y' is set as the density estimate, where the smoothed curved is faced to, right ('positive') or left ('negative') as vertical layout; up ('positive') or down ('negative') as horizontal layout?

adjust — adjust the proportional maximum height of the estimate (density, histogram, ...).

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

orientation — The orientation of the layer. The default (NA) automatically determines the orientation from the aesthetic mapping. In the rare event that this fails it can be given explicitly by setting orientation to either "x" or "y". See the *Orientation* section for more detail.

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

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

geom — The geometric object to use display the data

axes.position	A numerical vector to determine the axes sequence position; the length should be the same with the length of axes.sequence (or mapping aesthetics, see examples).
scaling	one of 'variable', 'data', 'observation' or 'none' to specify how the data is scaled.
bw	The smoothing bandwidth to be used. If numeric, the standard deviation of the smoothing kernel. If character, a rule to choose the bandwidth, as listed in stats::bw.nrd().
kernel	Kernel. See list of available kernels in density().
n	number of equally spaced points at which the density is to be estimated, should be a power of two, see density() for details
trim	If FALSE, the default, each density is computed on the full range of the data. If TRUE, each density is computed over the range of that group: this typically means the estimated x values will not line-up, and hence you won't be able to stack density values. This parameter only matters if you are displaying multiple densities in one plot or if you are manually adjusting the scale limits.

See Also

geom_density_, geom_serialaxes, geom_serialaxes_quantile, geom_serialaxes_hist

Examples

```
p <- ggplot(iris, mapping = aes(Sepal.Length = Sepal.Length,
                                Sepal.Width = Sepal.Width,
                                Petal.Length = Petal.Length,
                                Petal.Width = Petal.Width,
                                colour = Species,
                                fill = Species)) +
       geom_serialaxes(alpha = 0.2) +
       geom_serialaxes_density(alpha = 0.5) +
       scale_x_continuous(breaks = 1:4,
                          labels = colnames(iris)[-5]) +
       scale_y_continuous(labels = NULL) +
       xlab("variable") +
       ylab("") +
       theme(axis.text.x = element_text(angle = 45, vjust = 0.5))
p
```

---

geom_serialaxes_glyph *Add serial axes glyphs on scatter plot*

Description

To visualize high dimensional data on scatterplot. Each point glyph is surrounded by a serial axes (parallel axes or radial axes) object.

Usage

```
geom_serialaxes_glyph(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  ...,
  serialaxes.data,
  axes.sequence = character(0L),
  scaling = c("variable", "data", "observation", "none"),
  axes.layout = c("parallel", "radial"),
  andrews = FALSE,
  show.axes = FALSE,
  show.enclosing = FALSE,
  linewidth = 1,
  axescolour = "black",
  bboxcolour = "black",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by aes() or aes_(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x,10)).
stat	The statistical transformation to use on the data for this layer, as a string.
position	Position adjustment, either as a string, or the result of a call to a position adjustment function.
...	Other arguments passed on to layer(). These are often aesthetics, used to set an aesthetic to a fixed value, like colour = "red" or size = 3. They may also be parameters to the paired geom/stat.
serialaxes.data	a serial axes numerical data set. If not provided, geom_point() will be called.
axes.sequence	A vector to define the axes sequence. In serial axes coordinate, the sequence can be either determined in mapping (function aes()) or by axes.sequence.

	The only difference is that the `mapping` aesthetics will omit the duplicated axes (check examples in `geom_serialaxes`).
`scaling`	one of 'variable', 'data', 'observation' or 'none' to specify how the data is scaled.
`axes.layout`	either "radial" or "parallel"
`andrews`	Logical; Andrew's plot (a 'Fourier' transformation)
`show.axes`	boolean to indicate whether axes should be shown or not
`show.enclosing`	boolean to indicate whether enclosing should be shown or not
`linewidth`	line width of the "glyph" object
`axescolour`	axes color
`bboxcolour`	bounding box color
`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 `geom` layer

Aesthetics

geom_..._glyph() understands the following aesthetics (required aesthetics are in bold):

- **x**
- **y**
- alpha
- colour
- fill
- group
- shape
- size
- stroke
- linetype

See Also

`geom_polygon_glyph`, `geom_image_glyph`

Examples

```
# serial axes glyph
p <- ggplot(data = iris,
            mapping = aes(x = Sepal.Length, y = Sepal.Width, colour = Species)) +
  geom_serialaxes_glyph(serialaxes.data = iris[, -5],
                        axes.layout = "radial")
p
```

`geom_serialaxes_hist` *Histogram for "widens" data under serial axes coordinate*

Description

Computes and draws histogram on serial axes coordinate for each non-aesthetics component defined in the mapping `aes()`.

Usage

```
geom_serialaxes_hist(
  mapping = NULL,
  data = NULL,
  stat = "serialaxes_hist",
  position = "stack_",
  ...,
  axes.sequence = character(0L),
  axes.position = NULL,
  merge = TRUE,
  scale.y = c("data", "variable"),
  as.mix = TRUE,
  positive = TRUE,
  adjust = 0.9,
  na.rm = FALSE,
  orientation = NA,
  show.legend = NA,
  inherit.aes = TRUE
)

stat_serialaxes_hist(
  mapping = NULL,
  data = NULL,
  geom = "serialaxes_hist",
  position = "stack_",
  ...,
  axes.sequence = character(0L),
  scaling = c("variable", "observation", "data", "none"),
  axes.position = NULL,
  binwidth = NULL,
```

```
  bins = NULL,
  center = NULL,
  boundary = NULL,
  breaks = NULL,
  closed = c("right", "left"),
  pad = FALSE,
  width = NULL,
  na.rm = FALSE,
  orientation = NA,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by aes() or aes_(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x,10)).
stat	The statistical transformation to use on the data for this layer, as a string.
position	Position adjustment, either as a string, or the result of a call to a position adjustment function.
...	Other arguments passed on to layer(). These are often aesthetics, used to set an aesthetic to a fixed value, like colour = "red" or size = 3. They may also be parameters to the paired geom/stat.
axes.sequence	A vector to define the axes sequence. In serial axes coordinate, the sequence can be either determined in mapping (function aes()) or by axes.sequence. The only difference is that the mapping aesthetics will omit the duplicated axes (check examples in geom_serialaxes).
axes.position	A numerical vector to determine the axes sequence position; the length should be the same with the length of axes.sequence (or mapping aesthetics, see examples).
merge	Should axes.sequence be merged with mapping aesthetics as a single mapping uneval object?
scale.y	one of 'data', 'variable' to specify.

Type	Description
data (default)	The density estimates are scaled by the whole data set
variable	The density estimates are scaled by each variable

If the scale.y is "data", it is meaningful to compare the density (shape and area) across all groups; else it is only meaningful to compare the density under each variable.

as.mix — Logical. Under each variable, if as.mix = TRUE, the sum of the density estimate area is mixed and scaled to maximum 1. The area of each group is proportional to its own count; if as.mix = FALSE the area of each group is the same, with maximum 1.

positive — If 'y' is set as the density estimate, where the smoothed curved is faced to, right ('positive') or left ('negative') as vertical layout; up ('positive') or down ('negative') as horizontal layout?

adjust — adjust the proportional maximum height of the estimate (density, histogram, ...).

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

orientation — The orientation of the layer. The default (NA) automatically determines the orientation from the aesthetic mapping. In the rare event that this fails it can be given explicitly by setting orientation to either "x" or "y". See the *Orientation* section for more detail.

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

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

geom — The geometric object to use display the data

scaling — one of 'variable', 'data', 'observation' or 'none' to specify how the data is scaled.

binwidth — The width of the bins. Can be specified as a numeric value or as a function that calculates width from unscaled x. Here, "unscaled x" refers to the original x values in the data, before application of any scale transformation. When specifying a function along with a grouping structure, the function will be called once per group. The default is to use the number of bins in bins, covering the range of the data. You should always override this value, exploring multiple widths to find the best to illustrate the stories in your data.

The bin width of a date variable is the number of days in each time; the bin width of a time variable is the number of seconds.

bins — Number of bins. Overridden by binwidth. Defaults to 30.

center — bin position specifiers. Only one, center or boundary, may be specified for a single plot. center specifies the center of one of the bins. boundary specifies the boundary between two bins. Note that if either is above or below the range of the data, things will be shifted by the appropriate integer multiple of binwidth. For example, to center on integers use binwidth = 1 and center = 0, even if 0 is outside the range of the data. Alternatively, this same alignment can be specified with binwidth = 1 and boundary = 0.5, even if 0.5 is outside the range of the data.

boundary	bin position specifiers. Only one, center or boundary, may be specified for a single plot. center specifies the center of one of the bins. boundary specifies the boundary between two bins. Note that if either is above or below the range of the data, things will be shifted by the appropriate integer multiple of binwidth. For example, to center on integers use binwidth = 1 and center = 0, even if 0 is outside the range of the data. Alternatively, this same alignment can be specified with binwidth = 1 and boundary = 0.5, even if 0.5 is outside the range of the data.
breaks	Alternatively, you can supply a numeric vector giving the bin boundaries. Overrides binwidth, bins, center, and boundary.
closed	One of "right" or "left" indicating whether right or left edges of bins are included in the bin.
pad	If TRUE, adds empty bins at either end of x. This ensures frequency polygons touch 0. Defaults to FALSE.
width	Bar width. By default, set to 90% of the resolution of the data.

See Also

geom_hist_, geom_serialaxes, geom_serialaxes_quantile, geom_serialaxes_density

Examples

```
p <- ggplot(iris, mapping = aes(Sepal.Length = Sepal.Length,
                                Sepal.Width = Sepal.Width,
                                Petal.Length = Petal.Length,
                                Petal.Width = Petal.Width,
                                colour = Species,
                                fill = Species)) +
            geom_serialaxes(alpha = 0.2) +
            geom_serialaxes_hist(alpha = 0.5) +
            scale_x_continuous(breaks = 1:4,
                               labels = colnames(iris)[-5]) +
            scale_y_continuous(labels = NULL) +
            xlab("variable") +
            ylab("") +
            theme(axis.text.x = element_text(angle = 45, vjust = 0.5))
p
```

---

geom_serialaxes_quantile

Quantile layer for serial axes coordinate

Description

Draw a quantile layer for serial axes coordinate. Don't be confused with geom_quantile() which is a quantile regression. See examples.

Usage

```
geom_serialaxes_quantile(
  mapping = NULL,
  data = NULL,
  stat = "serialaxes",
  position = "identity",
  ...,
  axes.sequence = character(0L),
  merge = TRUE,
  na.rm = FALSE,
  orientation = NA,
  show.legend = NA,
  inherit.aes = TRUE
)

stat_serialaxes_quantile(
  mapping = NULL,
  data = NULL,
  geom = "serialaxes_quantile",
  position = "identity",
  ...,
  axes.sequence = character(0L),
  merge = TRUE,
  quantiles = seq(0, 1, 0.25),
  scaling = c("variable", "observation", "data", "none"),
  axes.position = NULL,
  na.rm = FALSE,
  orientation = NA,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

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

position	Position adjustment, either as a string, or the result of a call to a position adjustment function.
...	Other arguments passed on to layer(). These are often aesthetics, used to set an aesthetic to a fixed value, like colour = "red" or size = 3. They may also be parameters to the paired geom/stat.
axes.sequence	A vector to define the axes sequence. In serial axes coordinate, the sequence can be either determined in mapping (function aes()) or by axes.sequence. The only difference is that the mapping aesthetics will omit the duplicated axes (check examples in geom_serialaxes).
merge	Should axes.sequence be merged with mapping aesthetics as a single mapping uneval object?
na.rm	If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.
orientation	The orientation of the layer. The default (NA) automatically determines the orientation from the aesthetic mapping. In the rare event that this fails it can be given explicitly by setting orientation to either "x" or "y". See the *Orientation* section for more detail.
show.legend	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.
inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. borders().
geom	The geometric object to use display the data
quantiles	numeric vector of probabilities with values in [0,1]. (Values up to 2e-14 outside that range are accepted and moved to the nearby endpoint.)
scaling	one of 'variable', 'data', 'observation' or 'none' to specify how the data is scaled.
axes.position	A numerical vector to determine the axes sequence position; the length should be the same with the length of axes.sequence (or mapping aesthetics, see examples).

See Also

geom_density_, geom_serialaxes, geom_serialaxes_density, geom_serialaxes_hist

Examples

```
# lower quantile, median and upper quantile
p <- ggplot(iris, mapping = aes(Sepal.Length = Sepal.Length,
                                Sepal.Width = Sepal.Width,
                                Petal.Length = Petal.Length,
                                Petal.Width = Petal.Width)) +
       geom_serialaxes(stat = "dotProduct") +
       geom_serialaxes_quantile(stat = "dotProduct",
                                quantiles = c(0.25, 0.5, 0.75),
                                colour = c("red", "blue", "green"), size = 2)
p
```

get_scaledData *scale data*

Description

It is mainly used in serial axes

Usage

```
get_scaledData(
  data,
  sequence = NULL,
  scaling = c("variable", "data", "observation", "none"),
  displayOrder = NULL,
  reserve = FALSE,
  as.data.frame = FALSE
)
```

Arguments

data	A data frame
sequence	vector with variable names that defines the axes sequence. If NULL, it will be set as the column names automatically.
scaling	one of 'variable', 'data', 'observation' or 'none' to specify how the data is scaled.
displayOrder	the order of the display
reserve	If TRUE, return the variables not shown in sequence as well; else only return the variables defined in sequence.
as.data.frame	Return a matrix or a data.frame

polygon_glyph *Polygon glyph coordinates*

Description

polygon coordinates scaled to (0, 1)

Usage

```
x_star

y_star

x_cross
```

y_cross

x_hexagon

y_hexagon

x_airplane

y_airplane

x_maple

y_maple

Format

An object of class numeric of length 10.

An object of class numeric of length 10.

An object of class numeric of length 12.

An object of class numeric of length 12.

An object of class numeric of length 6.

An object of class numeric of length 6.

An object of class numeric of length 32.

An object of class numeric of length 32.

An object of class numeric of length 26.

An object of class numeric of length 26.

See Also

geom_polygon_glyph

Examples

```
if(requireNamespace("grid")) {
  library(grid)
  grid.newpage()
  grid.polygon(x=(x_star + 1)/2,
               y=(y_star + 1)/2)
  grid.newpage()
  grid.polygon(x=(x_cross + 1)/2,
               y=(y_cross + 1)/2)
  grid.newpage()
  grid.polygon(x=(x_hexagon + 1)/2,
               y=(y_hexagon + 1)/2)
  grid.newpage()
  grid.polygon(x=(x_airplane + 1)/2,
               y=(y_airplane + 1)/2)
```

```
  grid.newpage()
  grid.polygon(x=(x_maple + 1)/2,
               y=(y_maple + 1)/2)
}
```

`Position-ggproto` *Base Position ggproto classes for ggplot2*

Description

All `position_` functions (like `position_dodge`) return a `Position` object (like `PositionDodge`). The `Position` object is responsible for adjusting the position of overlapping geoms. The way that the `position_` functions work is slightly different from the `geom_` and `stat_` functions, because a `position_` function actually "instantiates" the `Position` object by creating a descendant, and returns that. Each of the `Position` objects is a `ggproto` object, descended from the top-level `Position`.

Usage

```
PositionDodge_

PositionDodge2_

PositionIdentity_

PositionStack_

PositionFill_
```

Format

An object of class `PositionDodge_` (inherits from `PositionDodge`, `Position`, `ggproto`, `gg`) of length 2.

An object of class `PositionDodge2_` (inherits from `PositionDodge2`, `PositionDodge`, `Position`, `ggproto`, `gg`) of length 2.

An object of class `PositionIdentity_` (inherits from `PositionIdentity`, `Position`, `ggproto`, `gg`) of length 3.

An object of class `PositionStack_` (inherits from `PositionStack`, `Position`, `ggproto`, `gg`) of length 3.

An object of class `PositionFill_` (inherits from `PositionStack_`, `PositionStack`, `Position`, `ggproto`, `gg`) of length 2.

position_dodge_ *Dodge overlapping objects side-to-side*

Description

Dodging preserves the vertical position of an geom while adjusting the horizontal position. position_dodge_() dodges bars side by side but conditional on locations.

Usage

```
position_dodge_(width = NULL, preserve = c("total", "single"))

position_dodge2_(
  width = NULL,
  preserve = c("total", "single"),
  padding = 0.1,
  reverse = FALSE
)
```

Arguments

width	Dodging width, when different to the width of the individual elements. This is useful when you want to align narrow geoms with wider geoms. See the examples.
preserve	Should dodging preserve the total width of all elements at a position, or the width of a single element?
padding	Padding between elements at the same position. Elements are shrunk by this proportion to allow space between them. Defaults to 0.1.
reverse	If TRUE, will reverse the default stacking order. This is useful if you're rotating both the plot and legend.

Details

It is built based on position_dodge, but used for multiple locations, such as geom_hist_() or geom_density_(). Check examples to see the difference.

See Also

See geom_hist_ and geom_serialaxes_hist for more examples.

Other position adjustments for multiple locations: position_identity_, position_stack_, position_fill_

Parent: position_dodge

Examples

```
if(require(dplyr)) {
p <- iris %>%
  tidyr::pivot_longer(cols = -Species,
                      names_to = "Outer sterile whorls",
                      values_to = "values") %>%
  ggplot(data,
         mapping = aes(x = `Outer sterile whorls`,
                       y = values,
                       fill = Species))

p + geom_hist_(position = position_dodge_())
}


# all bins are shifted on the left
p +
  geom_hist_(position = position_dodge())
```

---

position_identity_ *Don't adjust position*

Description

Don't adjust position

Usage

```
position_identity_()
```

See Also

Other position adjustments for multiple locations: position_stack_, position_fill_, position_dodge_, position_dodge2_

---

position_stack_ *Stack overlapping objects on top of each another*

Description

position_stack_ stacks bars on top of each other, conditional on locations.

Usage

```
position_stack_(vjust = 1, reverse = FALSE)

position_fill_(vjust = 1, reverse = FALSE)
```

Arguments

vjust	Vertical adjustment for geoms that have a position (like points or lines), not a dimension (like bars or areas). Set to `0` to align with the bottom, `0.5` for the middle, and `1` (the default) for the top.
reverse	If `TRUE`, will reverse the default stacking order. This is useful if you're rotating both the plot and legend.

Details

It is built based on `position_stack`, but used for multiple locations, such as `geom_hist_` or `geom_density_`. Rather than stack everything on top of each other, `position_stack_` stacks bars based on locations. Check examples to see the difference.

See Also

See `geom_hist_`, `geom_density_`, `geom_serialaxes_density` and `geom_serialaxes_hist` for more examples.

Other position adjustments for multiple locations: `position_identity_`, `position_dodge_`, `position_dodge2_`

Parent: `position_stack`

Examples

```
p <- ggplot(iris,
       mapping = aes(Sepal.Length = Sepal.Length,
                     Sepal.Width = Sepal.Width,
                     Petal.Length = Petal.Length,
                     Petal.Width = Petal.Width,
                     colour = Species))
p +
 geom_serialaxes_density(position = position_stack_())


p +
  geom_serialaxes_density(position = position_stack())
```

Stat-ggproto *Base Stat ggproto classes for ggplot2*

Description

All stat_ functions (like stat_bin()) return a layer that contains a Stat object (like StatBin). The Stat object is responsible for rendering the data in the plot. Each of the Stat objects is a ggproto object, descended from the top-level Stat, and each implements various methods and fields.

Usage

```
StatDensity_

StatHist_

StatBin_

StatCount_

StatSerialaxesDensity

StatSerialaxesHist

StatSerialaxes

StatDotProduct
```

Format

An object of class StatDensity_ (inherits from StatDensity, Stat, ggproto, gg) of length 4.

An object of class StatHist_ (inherits from StatBin, Stat, ggproto, gg) of length 4.

An object of class StatBin_ (inherits from StatHist_, StatBin, Stat, ggproto, gg) of length 2.

An object of class StatCount_ (inherits from StatHist_, StatBin, Stat, ggproto, gg) of length 2.

An object of class StatSerialaxesDensity (inherits from StatDensity, Stat, ggproto, gg) of length 4.

An object of class StatSerialaxesHist (inherits from StatBin, Stat, ggproto, gg) of length 4.

An object of class StatSerialaxes (inherits from Stat, ggproto, gg) of length 6.

An object of class StatDotProduct (inherits from StatSerialaxes, Stat, ggproto, gg) of length 4.

Index