Package 'ggalluvial'

July 22, 2025

Type Package

Title Alluvial Plots in 'ggplot2'

Version 0.12.5

Maintainer Jason Cory Brunson <cornelioid@gmail.com>

Description Alluvial plots use variable-width ribbons and stacked bar plots to represent multi-dimensional or repeated-measures data with categorical or ordinal variables; see Riehmann, Hanfler, and Froehlich (2005) <doi:10.1109/INFVIS.2005.1532152> and Rosvall and Bergstrom (2010) <doi:10.1371/journal.pone.0008694>.
Alluvial plots are statistical graphics in the sense of Wilkinson (2006) <doi:10.1007/0-387-28695-0>; they share elements with Sankey diagrams and parallel sets plots but are uniquely determined from the data and a small set of parameters. This package extends Wickham's (2010) <doi:10.1198/jcgs.2009.07098> layered grammar of graphics to generate alluvial plots from tidy data.

Depends R (>= 3.6), ggplot2 (>= 2.2)

Imports stats, dplyr (>= 0.7), tidyr (>= 0.7), lazyeval, rlang, tidyselect

Suggests grid, alluvial, testthat, knitr, rmarkdown, babynames, sessioninfo, ggrepel, shiny (>= 1.4.0.2), htmltools, sp (>= 1.4-0), ggfittext (>= 0.6), vdiffr (>= 0.2)

License GPL-3

LazyData true

URL http://corybrunson.github.io/ggalluvial/

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

VignetteBuilder knitr

RoxygenNote 7.2.3

Encoding UTF-8

NeedsCompilation no

Author Jason Cory Brunson [aut, cre], Quentin D. Read [aut]

Repository CRAN

Date/Publication 2023-02-22 09:50:02 UTC

Contents

`alluvial-data` *Check for alluvial structure and convert between alluvial formats*

Description

Alluvial plots consist of multiple horizontally-distributed columns (axes) representing factor variables, vertical divisions (strata) of these axes representing these variables' values; and splines (alluvial flows) connecting vertical subdivisions (lodes) within strata of adjacent axes representing subsets or amounts of observations that take the corresponding values of the corresponding variables. This function checks a data frame for either of two types of alluvial structure:

Usage

```
is_lodes_form(
  data,
  key,
  value,
  id,
  weight = NULL,
  site = NULL,
  logical = TRUE,
  silent = FALSE
)

is_alluvia_form(
  data,
  ...,
```

```
  axes = NULL,
  weight = NULL,
  logical = TRUE,
  silent = FALSE
)

to_lodes_form(
  data,
  ...,
  axes = NULL,
  key = "x",
  value = "stratum",
  id = "alluvium",
  diffuse = FALSE,
  discern = FALSE
)

to_alluvia_form(data, key, value, id, distill = FALSE)
```

Arguments

data	A data frame.
key, value, id	In to_lodes_form, handled as in tidyr::gather() and used to name the new axis (key), stratum (value), and alluvium (identifying) variables. In to_alluvia_form, handled as in tidyr::spread() and used to identify the fields of data to be used as the axis (key), stratum (value), and alluvium (identifying) variables.
weight	Optional field of data, handled using rlang::enquo(), to be used as heights or depths of the alluvia or lodes.
site	Optional vector of fields of data, handled using rlang::enquos(), to be used to group rows before testing for duplicate and missing id-axis pairings. Variables intended for faceting should be passed to site.
logical	Defunct. Whether to return a logical value or a character string indicating the type of alluvial structure ("none", "lodes", or "alluvia").
silent	Whether to print messages.
...	Used in is_alluvia_form and to_lodes_form as in dplyr::select() to determine axis variables, as an alternative to axes. Ignored when axes is provided.
axes	In *_alluvia_form, handled as in dplyr::select() and used to identify the field(s) of data to be used as axes.
diffuse	Fields of data, handled using tidyselect::vars_select(), to merge into the reshapen data by id. They must be a subset of the axis variables. Alternatively, a logical value indicating whether to merge all (TRUE) or none (FALSE) of the axis variables.
discern	Logical value indicating whether to suffix values of the variables used as axes that appear at more than one variable in order to distinguish their factor levels. This forces the levels of the combined factor variable value to be in the order of the axes.

`distill` A logical value indicating whether to include variables, other than those passed to `key` and `value`, that vary within values of `id`. Alternatively, a function (or its name) to be used to distill each such variable to a single value. In addition to existing functions, `distill` accepts the character values `"first"` (used if `distill` is `TRUE`), `"last"`, and `"most"` (which returns the first modal value).

Details

- One row per **lode**, wherein each row encodes a subset or amount of observations having a specific profile of axis values, a `key` field encodes the axis, a `value` field encodes the value within each axis, and a `id` column identifies multiple lodes corresponding to the same subset or amount of observations. `is_lodes_form` tests for this structure.
- One row per **alluvium**, wherein each row encodes a subset or amount of observations having a specific profile of axis values and a set `axes` of fields encodes its values at each axis variable. `is_alluvia_form` tests for this structure.

`to_lodes_form` takes a data frame with several designated variables to be used as axes in an alluvial plot, and reshapes the data frame so that the axis variable names constitute a new factor variable and their values comprise another. Other variables' values will be repeated, and a row-grouping variable can be introduced. This function invokes `tidyr::gather()`.

`to_alluvia_form` takes a data frame with axis and axis value variables to be used in an alluvial plot, and reshape the data frame so that the axes constitute separate variables whose values are given by the value variable. This function invokes `tidyr::spread()`.

See Also

Other alluvial data manipulation: `self-adjoin`

Examples

```
# Titanic data in alluvia format
titanic_alluvia <- as.data.frame(Titanic)
head(titanic_alluvia)
is_alluvia_form(titanic_alluvia,
                weight = "Freq")
# Titanic data in lodes format
titanic_lodes <- to_lodes_form(titanic_alluvia,
                               key = "x", value = "stratum", id = "alluvium",
                               axes = 1:4)
head(titanic_lodes)
is_lodes_form(titanic_lodes,
              key = "x", value = "stratum", id = "alluvium",
              weight = "Freq")
# again in lodes format, this time diffusing the `Class` variable
titanic_lodes2 <- to_lodes_form(titanic_alluvia,
                                key = variable, value = value,
                                id = cohort,
                                1:3, diffuse = Class)
head(titanic_lodes2)
is_lodes_form(titanic_lodes2,
              key = variable, value = value, id = cohort,
```

```
              weight = Freq)
# use `site` to separate data before lode testing
is_lodes_form(titanic_lodes2,
              key = variable, value = value, id = Class,
              weight = Freq)
is_lodes_form(titanic_lodes2,
              key = variable, value = value, id = Class,
              weight = Freq, site = cohort)

# curriculum data in lodes format
data(majors)
head(majors)
is_lodes_form(majors,
              key = "semester", value = "curriculum", id = "student")
# curriculum data in alluvia format
majors_alluvia <- to_alluvia_form(majors,
                                  key = "semester", value = "curriculum",
                                  id = "student")
head(majors_alluvia)
is_alluvia_form(majors_alluvia, tidyselect::starts_with("CURR"))

# distill variables that vary within `id` values
set.seed(1)
majors$hypo_grade <- LETTERS[sample(5, size = nrow(majors), replace = TRUE)]
majors_alluvia2 <- to_alluvia_form(majors,
                                   key = "semester", value = "curriculum",
                                   id = "student",
                                   distill = "most")
head(majors_alluvia2)

# options to distinguish strata at different axes
gg <- ggplot(majors_alluvia,
             aes(axis1 = CURR1, axis2 = CURR7, axis3 = CURR13))
gg +
  geom_alluvium(aes(fill = as.factor(student)), width = 2/5, discern = TRUE) +
  geom_stratum(width = 2/5, discern = TRUE) +
  geom_text(stat = "stratum", discern = TRUE, aes(label = after_stat(stratum)))
gg +
  geom_alluvium(aes(fill = as.factor(student)), width = 2/5, discern = FALSE) +
  geom_stratum(width = 2/5, discern = FALSE) +
  geom_text(stat = "stratum", discern = FALSE, aes(label = after_stat(stratum)))
# warning when inappropriate
ggplot(majors[majors$semester %in% paste0("CURR", c(1, 7, 13)), ],
       aes(x = semester, stratum = curriculum, alluvium = student,
           label = curriculum)) +
  geom_alluvium(aes(fill = as.factor(student)), width = 2/5, discern = TRUE) +
  geom_stratum(width = 2/5, discern = TRUE) +
  geom_text(stat = "stratum", discern = TRUE)
```

---

geom_alluvium *Alluvia across strata*

Description

geom_alluvium receives a dataset of the horizontal (x) and vertical (y, ymin, ymax) positions of the **lodes** of an alluvial plot, the intersections of the alluvia with the strata. It plots both the lodes themselves, using geom_lode(), and the flows between them, using geom_flow().

Usage

```
geom_alluvium(
  mapping = NULL,
  data = NULL,
  stat = "alluvium",
  position = "identity",
  width = 1/3,
  knot.pos = 1/4,
  knot.prop = TRUE,
  curve_type = NULL,
  curve_range = NULL,
  segments = NULL,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE,
  ...
)

data_to_alluvium(
  data,
  knot.prop = TRUE,
  curve_type = "spline",
  curve_range = NULL,
  segments = NULL
)
```

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; override the default.

position	Position adjustment, either as a string naming the adjustment (e.g. "jitter" to use position_jitter), or the result of a call to a position adjustment function. Use the latter if you need to change the settings of the adjustment.
width	Numeric; the width of each stratum, as a proportion of the distance between axes. Defaults to 1/3.
knot.pos	The horizontal distance of x-spline knots from each stratum (width/2 from its axis), either (if knot.prop = TRUE, the default) as a proportion of the length of the x-spline, i.e. of the gap between adjacent strata, or (if knot.prop = FALSE) on the scale of the x direction.
knot.prop	Logical; whether to interpret knot.pos as a proportion of the length of each flow (the default), rather than on the x scale.
curve_type	Character; the type of curve used to produce flows. Defaults to "xspline" and can be alternatively set to one of "linear", "cubic", "quintic", "sine", "arctangent", and "sigmoid". "xspline" produces approximation splines using 4 points per curve; the alternatives produce interpolation splines between points along the graphs of functions of the associated type. See the **Curves** section.
curve_range	For alternative curve_types based on asymptotic functions, the value along the asymptote at which to truncate the function to obtain the shape that will be scaled to fit between strata. See the **Curves** section.
segments	The number of segments to be used in drawing each alternative curve (each curved boundary of each flow). If less than 3, will be silently changed to 3.
na.rm	Logical: if FALSE, the default, NA lodes are not included; if TRUE, NA lodes constitute a separate category, plotted in grey (regardless of the color scheme).
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

The helper function data_to_alluvium() takes internal **ggplot2** data (mapped aesthetics) and curve parameters for a single alluvium as input and returns a data frame of x, y, and shape used by grid::xsplineGrob() to render the alluvium.

Aesthetics

geom_alluvium, geom_flow, geom_lode, and geom_stratum understand the following aesthetics (required aesthetics are in bold):

- x
- y
- ymin

- ymax
- alpha
- colour
- fill
- linetype
- size
- group

group is used internally; arguments are ignored.

Curves

By default, geom_alluvium() and geom_flow() render flows between lodes as filled regions between parallel x-splines. These graphical elements, generated using grid::xsplineGrob(), are parameterized by the relative location of the knot (knot.pos). They are quick to render and clear to read, but users may prefer plots that use differently-shaped ribbons.

A variety of such options are documented at, e.g., this easing functions cheat sheet and this blog post by Jeffrey Shaffer. Easing functions are not (yet) used in ggalluvial, but several alternative curves are available. Each is encoded as a continuous, increasing, bijective function from the unit interval $[0, 1]$ to itself, and each is rescaled so that its endpoints meet the corresponding lodes. They are rendered piecewise-linearly, by default using segments = 48. Summon each curve type by passing one of the following strings to curve_type:

- "linear": $f(x) = x$, the unique degree-1 polynomial that takes 0 to 0 and 1 to 1
- "cubic": $f(x) = 3x^2 - 2x^3$, the unique degree-3 polynomial that also is flat at both endpoints
- "quintic": $f(x) = 10x^3 - 15x^4 + 6x^5$, the unique degree-5 polynomial that also has zero curvature at both endpoints
- "sine": the unique sinusoidal function that is flat at both endpoints
- "arctangent": the inverse tangent function, scaled and re-centered to the unit interval from the interval centered at zero with radius curve_range
- "sigmoid": the sigmoid function, scaled and re-centered to the unit interval from the interval centered at zero with radius curve_range

Only the (default) "xspline" option uses the knot.* parameters, while only the alternative curves use the segments parameter, and only "arctangent" and "sigmoid" use the curve_range parameter. (Both are ignored if not needed.) Larger values of curve_range result in greater compression and steeper slopes. The NULL default will be changed to 2+sqrt(3) for "arctangent" and to 6 for "sigmoid".

These package-specific options set global values for curve_type, curve_range, and segments that will be defaulted to when not manually set:

- ggalluvial.curve_type: defaults to "xspline".
- ggalluvial.curve_range: defaults to NA, which triggers the curve-specific default values.
- ggalluvial.segments: defaults to 48L.

See base::options() for how to use options.

Defunct parameters

The previously defunct parameters `axis_width` and `ribbon_bend` have been discontinued. Use `width` and `knot.pos` instead.

See Also

`ggplot2::layer()` for additional arguments and `stat_alluvium()` and `stat_flow()` for the corresponding stats.

Other alluvial geom layers: `geom_flow()`, `geom_lode()`, `geom_stratum()`

Examples

```
# basic
ggplot(as.data.frame(Titanic),
       aes(y = Freq,
           axis1 = Class, axis2 = Sex, axis3 = Age,
           fill = Survived)) +
  geom_alluvium() +
  scale_x_discrete(limits = c("Class", "Sex", "Age"))

gg <- ggplot(alluvial::Refugees,
             aes(y = refugees, x = year, alluvium = country))
# time series bump chart (quintic flows)
gg + geom_alluvium(aes(fill = country, colour = country),
                   width = 1/4, alpha = 2/3, decreasing = FALSE,
                   curve_type = "sigmoid")
# time series line plot of refugees data, sorted by country
gg + geom_alluvium(aes(fill = country, colour = country),
                   decreasing = NA, width = 0, knot.pos = 0)


# irregular spacing between axes of a continuous variable
refugees_sub <- subset(alluvial::Refugees, year %in% c(2003, 2005, 2010, 2013))
gg <- ggplot(data = refugees_sub,
             aes(x = year, y = refugees, alluvium = country)) +
  theme_bw() +
  scale_fill_brewer(type = "qual", palette = "Set3")
# proportional knot positioning (default)
gg +
  geom_alluvium(aes(fill = country),
                alpha = .75, decreasing = FALSE, width = 1/2) +
  geom_stratum(aes(stratum = country), decreasing = FALSE, width = 1/2)
# constant knot positioning
gg +
  geom_alluvium(aes(fill = country),
                alpha = .75, decreasing = FALSE, width = 1/2,
                knot.pos = 1, knot.prop = FALSE) +
  geom_stratum(aes(stratum = country), decreasing = FALSE, width = 1/2)
# coarsely-segmented curves
gg +
  geom_alluvium(aes(fill = country),
```

```
                alpha = .75, decreasing = FALSE, width = 1/2,
                curve_type = "arctan", segments = 6) +
  geom_stratum(aes(stratum = country), decreasing = FALSE, width = 1/2)
# custom-ranged curves
gg +
  geom_alluvium(aes(fill = country),
                alpha = .75, decreasing = FALSE, width = 1/2,
                curve_type = "arctan", curve_range = 1) +
  geom_stratum(aes(stratum = country), decreasing = FALSE, width = 1/2)
```

`geom_flow` *Flows between lodes or strata*

Description

`geom_flow` receives a dataset of the horizontal (`x`) and vertical (`y`, `ymin`, `ymax`) positions of the **lodes** of an alluvial plot, the intersections of the alluvia with the strata. It reconfigures these into alluvial segments connecting pairs of corresponding lodes in adjacent strata and plots filled x-splines between each such pair, using a provided knot position parameter `knot.pos`, and filled rectangles at either end, using a provided `width`.

Usage

```
geom_flow(
  mapping = NULL,
  data = NULL,
  stat = "flow",
  position = "identity",
  width = 1/3,
  knot.pos = 1/4,
  knot.prop = TRUE,
  curve_type = NULL,
  curve_range = NULL,
  segments = NULL,
  aes.flow = "forward",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE,
  ...
)

positions_to_flow(
  x0,
  x1,
  ymin0,
  ymax0,
  ymin1,
```

```
  ymax1,
  kp0,
  kp1,
  knot.prop,
  curve_type,
  curve_range,
  segments
)
```

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; override the default.
position	Position adjustment, either as a string naming the adjustment (e.g. "jitter" to use position_jitter), or the result of a call to a position adjustment function. Use the latter if you need to change the settings of the adjustment.
width	Numeric; the width of each stratum, as a proportion of the distance between axes. Defaults to 1/3.
knot.pos	The horizontal distance of x-spline knots from each stratum (width/2 from its axis), either (if knot.prop = TRUE, the default) as a proportion of the length of the x-spline, i.e. of the gap between adjacent strata, or (if knot.prop = FALSE) on the scale of the x direction.
knot.prop	Logical; whether to interpret knot.pos as a proportion of the length of each flow (the default), rather than on the x scale.
curve_type	Character; the type of curve used to produce flows. Defaults to "xspline" and can be alternatively set to one of "linear", "cubic", "quintic", "sine", "arctangent", and "sigmoid". "xspline" produces approximation splines using 4 points per curve; the alternatives produce interpolation splines between points along the graphs of functions of the associated type. See the **Curves** section.
curve_range	For alternative curve_types based on asymptotic functions, the value along the asymptote at which to truncate the function to obtain the shape that will be scaled to fit between strata. See the **Curves** section.
segments	The number of segments to be used in drawing each alternative curve (each curved boundary of each flow). If less than 3, will be silently changed to 3.

aes.flow	Character; how inter-lode flows assume aesthetics from lodes. Options are "forward" and "backward".
na.rm	Logical: if FALSE, the default, NA lodes are not included; if TRUE, NA lodes constitute a separate category, plotted in grey (regardless of the color scheme).
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().
x0, x1, ymin0, ymax0, ymin1, ymax1, kp0, kp1	Numeric corner and knot position data for the ribbon of a single flow.

Details

The helper function positions_toflow() takes the corner and knot positions and curve parameters for a single flow as input and returns a data frame of x, y, and shape used by grid::xsplineGrob() to render the flow.

Aesthetics

geom_alluvium, geom_flow, geom_lode, and geom_stratum understand the following aesthetics (required aesthetics are in bold):

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

group is used internally; arguments are ignored.

Curves

By default, geom_alluvium() and geom_flow() render flows between lodes as filled regions between parallel x-splines. These graphical elements, generated using grid::xsplineGrob(), are parameterized by the relative location of the knot (knot.pos). They are quick to render and clear to read, but users may prefer plots that use differently-shaped ribbons.

A variety of such options are documented at, e.g., this easing functions cheat sheet and this blog post by Jeffrey Shaffer. Easing functions are not (yet) used in ggalluvial, but several alternative curves

are available. Each is encoded as a continuous, increasing, bijective function from the unit interval $[0, 1]$ to itself, and each is rescaled so that its endpoints meet the corresponding lodes. They are rendered piecewise-linearly, by default using `segments = 48`. Summon each curve type by passing one of the following strings to `curve_type`:

- `"linear"`: $f(x) = x$, the unique degree-1 polynomial that takes 0 to 0 and 1 to 1
- `"cubic"`: $f(x) = 3x^2 - 2x^3$, the unique degree-3 polynomial that also is flat at both endpoints
- `"quintic"`: $f(x) = 10x^3 - 15x^4 + 6x^5$, the unique degree-5 polynomial that also has zero curvature at both endpoints
- `"sine"`: the unique sinusoidal function that is flat at both endpoints
- `"arctangent"`: the inverse tangent function, scaled and re-centered to the unit interval from the interval centered at zero with radius `curve_range`
- `"sigmoid"`: the sigmoid function, scaled and re-centered to the unit interval from the interval centered at zero with radius `curve_range`

Only the (default) `"xspline"` option uses the `knot.*` parameters, while only the alternative curves use the `segments` parameter, and only `"arctangent"` and `"sigmoid"` use the `curve_range` parameter. (Both are ignored if not needed.) Larger values of `curve_range` result in greater compression and steeper slopes. The `NULL` default will be changed to `2+sqrt(3)` for `"arctangent"` and to `6` for `"sigmoid"`.

These package-specific options set global values for `curve_type`, `curve_range`, and `segments` that will be defaulted to when not manually set:

- `ggalluvial.curve_type`: defaults to `"xspline"`.
- `ggalluvial.curve_range`: defaults to `NA`, which triggers the curve-specific default values.
- `ggalluvial.segments`: defaults to `48L`.

See `base::options()` for how to use options.

Defunct parameters

The previously defunct parameters `axis_width` and `ribbon_bend` have been discontinued. Use `width` and `knot.pos` instead.

See Also

`ggplot2::layer()` for additional arguments and `stat_alluvium()` and `stat_flow()` for the corresponding stats.

Other alluvial geom layers: `geom_alluvium()`, `geom_lode()`, `geom_stratum()`

Examples

```
# use of strata and labels
ggplot(as.data.frame(Titanic),
       aes(y = Freq,
           axis1 = Class, axis2 = Sex, axis3 = Age)) +
  geom_flow() +
  scale_x_discrete(limits = c("Class", "Sex", "Age")) +
```

```
  geom_stratum() +
  geom_text(stat = "stratum", aes(label = after_stat(stratum))) +
  ggtitle("Alluvial plot of Titanic passenger demographic data")

# use of facets, with sigmoid flows
ggplot(as.data.frame(Titanic),
       aes(y = Freq,
           axis1 = Class, axis2 = Sex)) +
  geom_flow(aes(fill = Age), width = .4, curve_type = "quintic") +
  geom_stratum(width = .4) +
  geom_text(stat = "stratum", aes(label = after_stat(stratum)), size = 3) +
  scale_x_discrete(limits = c("Class", "Sex")) +
  facet_wrap(~ Survived, scales = "fixed")

# time series alluvia of WorldPhones data
wph <- as.data.frame(as.table(WorldPhones))
names(wph) <- c("Year", "Region", "Telephones")
ggplot(wph,
       aes(x = Year, alluvium = Region, y = Telephones)) +
  geom_flow(aes(fill = Region, colour = Region), width = 0)
# treat 'Year' as a number rather than as a factor
wph$Year <- as.integer(as.character(wph$Year))
ggplot(wph,
       aes(x = Year, alluvium = Region, y = Telephones)) +
  geom_flow(aes(fill = Region, colour = Region), width = 0)
# hold the knot positions fixed
ggplot(wph,
       aes(x = Year, alluvium = Region, y = Telephones)) +
  geom_flow(aes(fill = Region, colour = Region), width = 0, knot.prop = FALSE)


# rightward flow aesthetics for vaccine survey data, with cubic flows
data(vaccinations)
vaccinations$response <- factor(vaccinations$response,
                                rev(levels(vaccinations$response)))
# annotate with proportional counts
ggplot(vaccinations,
       aes(x = survey, stratum = response, alluvium = subject,
           y = freq, fill = response)) +
  geom_lode() + geom_flow(curve_type = "cubic") +
  geom_stratum(alpha = 0) +
  geom_text(stat = "stratum", aes(label = round(after_stat(prop), 3)))
# annotate fixed-width ribbons with counts
ggplot(vaccinations,
       aes(x = survey, stratum = response, alluvium = subject,
           weight = freq, fill = response)) +
  geom_lode() + geom_flow(curve_type = "cubic") +
  geom_stratum(alpha = 0) +
  geom_text(stat = "flow",
            aes(label = after_stat(n),
                hjust = (after_stat(flow) == "to")))
```

geom_lode *Lodes at intersections of alluvia and strata*

Description

`geom_alluvium` receives a dataset of the horizontal (`x`) and vertical (`y`, `ymin`, `ymax`) positions of the **lodes** of an alluvial plot, the intersections of the alluvia with the strata. It plots rectangles for these lodes of a provided `width`.

Usage

```
geom_lode(
  mapping = NULL,
  data = NULL,
  stat = "alluvium",
  position = "identity",
  width = 1/3,
  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; override the default.
`position`	Position adjustment, either as a string naming the adjustment (e.g. `"jitter"` to use `position_jitter`), or the result of a call to a position adjustment function. Use the latter if you need to change the settings of the adjustment.
`width`	Numeric; the width of each stratum, as a proportion of the distance between axes. Defaults to 1/3.
`na.rm`	Logical: if `FALSE`, the default, `NA` lodes are not included; if `TRUE`, `NA` lodes constitute a separate category, plotted in grey (regardless of the color scheme).

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().

Aesthetics

geom_alluvium, geom_flow, geom_lode, and geom_stratum understand the following aesthetics (required aesthetics are in bold):

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

group is used internally; arguments are ignored.

Defunct parameters

The previously defunct parameters axis_width and ribbon_bend have been discontinued. Use width and knot.pos instead.

See Also

ggplot2::layer() for additional arguments and stat_alluvium() and stat_stratum() for the corresponding stats.

Other alluvial geom layers: geom_alluvium(), geom_flow(), geom_stratum()

Examples

```
# one axis
ggplot(as.data.frame(Titanic),
       aes(y = Freq,
           axis = Class)) +
  geom_lode(aes(fill = Class, alpha = Survived)) +
  scale_x_discrete(limits = c("Class")) +
  scale_alpha_manual(values = c(.25, .75))
```

```
gg <- ggplot(as.data.frame(Titanic),
             aes(y = Freq,
                 axis1 = Class, axis2 = Sex, axis3 = Age,
                 fill = Survived))
# alluvia and lodes
gg + geom_alluvium() + geom_lode()
# lodes as strata
gg + geom_alluvium() +
  geom_stratum(stat = "alluvium")
```

`geom_stratum` *Strata at axes*

Description

`geom_stratum` receives a dataset of the horizontal (`x`) and vertical (`y`, `ymin`, `ymax`) positions of the strata of an alluvial plot. It plots rectangles for these strata of a provided `width`.

Usage

```
geom_stratum(
  mapping = NULL,
  data = NULL,
  stat = "stratum",
  position = "identity",
  show.legend = NA,
  inherit.aes = TRUE,
  width = 1/3,
  na.rm = FALSE,
  ...
)
```

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; override the default.

position	Position adjustment, either as a string naming the adjustment (e.g. "jitter" to use position_jitter), or the result of a call to a position adjustment function. Use the latter if you need to change the settings of the adjustment.
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().
width	Numeric; the width of each stratum, as a proportion of the distance between axes. Defaults to 1/3.
na.rm	Logical: if FALSE, the default, NA lodes are not included; if TRUE, NA lodes constitute a separate category, plotted in grey (regardless of the color scheme).
...	Additional arguments passed to ggplot2::layer().

Aesthetics

geom_alluvium, geom_flow, geom_lode, and geom_stratum understand the following aesthetics (required aesthetics are in bold):

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

group is used internally; arguments are ignored.

Defunct parameters

The previously defunct parameters axis_width and ribbon_bend have been discontinued. Use width and knot.pos instead.

See Also

ggplot2::layer() for additional arguments and stat_stratum() for the corresponding stat.

Other alluvial geom layers: geom_alluvium(), geom_flow(), geom_lode()

Examples

```
# full axis width
ggplot(as.data.frame(Titanic),
       aes(y = Freq,
           axis1 = Class, axis2 = Sex, axis3 = Age, axis4 = Survived)) +
  geom_stratum(width = 1) +
  geom_text(stat = "stratum", aes(label = after_stat(stratum))) +
  scale_x_discrete(limits = c("Class", "Sex", "Age", "Survived"))

# use of facets
ggplot(as.data.frame(Titanic),
       aes(y = Freq,
           axis1 = Class, axis2 = Sex)) +
  geom_flow(aes(fill = Survived)) +
  geom_stratum() +
  geom_text(stat = "stratum", aes(label = after_stat(stratum))) +
  scale_x_discrete(limits = c("Class", "Sex")) +
  facet_wrap(~ Age, scales = "free_y")
```

---

lode-guidance-functions

Lode guidance functions

Description

These functions control the order of lodes within strata in an alluvial diagram. They are invoked by `stat_alluvium()` and can be passed to the `lode.guidance` parameter.

Usage

```
lode_zigzag(n, i)

lode_zagzig(n, i)

lode_forward(n, i)

lode_rightward(n, i)

lode_backward(n, i)

lode_leftward(n, i)

lode_frontback(n, i)

lode_rightleft(n, i)

lode_backfront(n, i)

lode_leftright(n, i)
```

Arguments

n	Numeric, a positive integer
i	Numeric, a positive integer at most n

Details

Each function orders the numbers 1 through n, starting at index i. The choice of function made in stat_alluvium() determines the order in which the other axes contribute to the sorting of lodes within each index axis. After starting at i, the functions order the remaining axes as follows:

- zigzag: Zigzag outward from i, starting in the outward direction
- zigzag: Zigzag outward from i, starting in the inward direction
- forward: Increasing order (alias rightward)
- backward: Decreasing order (alias leftward)
- frontback: Proceed forward from i to n, then backward to 1 (alias rightleft)
- backfront: Proceed backward from i to 1, then forward to n (alias leftright)

An extended discussion of how strata and lodes are arranged in alluvial plots, including the effects of different lode guidance functions, can be found in the vignette "The Order of the Rectangles" via vignette("order-rectangles", package = "ggalluvial").

---

majors *Students' declared majors across several semesters*

Description

This data set follows the major curricula of 10 students across 8 academic semesters. Missing values indicate undeclared majors. The data were kindly contributed by Dario Bonaretti.

Format

A data frame with 80 rows and 3 variables:

student student identifier

semester character tag for odd-numbered semesters

curriculum declared major program

`self-adjoin` *Adjoin a dataset to itself*

Description

This function binds a dataset to itself along adjacent pairs of a `key` variable. It is invoked by `geom_flow()` to convert data in lodes form to something similar to alluvia form.

Usage

```
self_adjoin(
  data,
  key,
  by = NULL,
  link = NULL,
  keep.x = NULL,
  keep.y = NULL,
  suffix = c(".x", ".y")
)
```

Arguments

`data`	A data frame in lodes form (repeated measures data; see `alluvial-data`).
`key`	Column of `data` indicating sequential collection; handled as in `tidyr::spread()`.
`by`	Character vector of variables to self-adjoin by; passed to `dplyr::mutate-joins` functions.
`link`	Character vector of variables to adjoin. Will be replaced by pairs of variables suffixed by `suffix`.
`keep.x`, `keep.y`	Character vector of variables to associate with the first (respectively, second) copy of `data` after adjoining. These variables can overlap with each other but cannot overlap with `by` or `link`.
`suffix`	Suffixes to add to the adjoined `link` variables; passed to `dplyr::mutate-joins` functions.

Details

`self_adjoin` invokes `dplyr::mutate-joins` functions in order to convert a dataset with measures along a discrete `key` variable into a dataset consisting of column bindings of these measures (by any `by` variables) along adjacent values of `key`.

See Also

Other alluvial data manipulation: `alluvial-data`

Examples

```
# self-adjoin `majors` data
data(majors)
major_changes <- self_adjoin(majors, key = semester,
                             by = "student", link = c("semester", "curriculum"))
major_changes$change <- major_changes$curriculum.x == major_changes$curriculum.y
head(major_changes)

# self-adjoin `vaccinations` data
data(vaccinations)
vaccination_steps <- self_adjoin(vaccinations, key = survey, by = "subject",
                                 link = c("survey", "response"),
                                 keep.x = c("freq"))
head(vaccination_steps)
vaccination_steps <- self_adjoin(vaccinations, key = survey, by = "subject",
                                 link = c("survey", "response"),
                                 keep.x = c("freq"),
                                 keep.y = c("start_date", "end_date"))
head(vaccination_steps)
```

`stat_alluvium` *Alluvial positions*

Description

Given a dataset with alluvial structure, `stat_alluvium` calculates the centroids (`x` and `y`) and heights (`ymin` and `ymax`) of the lodes, the intersections of the alluvia with the strata. It leverages the `group` aesthetic for plotting purposes (for now).

Usage

```
stat_alluvium(
  mapping = NULL,
  data = NULL,
  geom = "alluvium",
  position = "identity",
  decreasing = NULL,
  reverse = NULL,
  absolute = NULL,
  discern = FALSE,
  negate.strata = NULL,
  aggregate.y = NULL,
  cement.alluvia = NULL,
  lode.guidance = NULL,
  lode.ordering = NULL,
  aes.bind = NULL,
  infer.label = FALSE,
  min.y = NULL,
```

```
  max.y = NULL,
  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)).
geom	The geometric object to use display the data; override the default.
position	Position adjustment, either as a string naming the adjustment (e.g. "jitter" to use position_jitter), or the result of a call to a position adjustment function. Use the latter if you need to change the settings of the adjustment.
decreasing	Logical; whether to arrange the strata at each axis in the order of the variable values (NA, the default), in ascending order of totals (largest on top, FALSE), or in descending order of totals (largest on bottom, TRUE).
reverse	Logical; if decreasing is NA, whether to arrange the strata at each axis in the reverse order of the variable values, so that they match the order of the values in the legend. Ignored if decreasing is not NA. Defaults to TRUE.
absolute	Logical; if some cases or strata are negative, whether to arrange them (respecting decreasing and reverse) using negative or absolute values of y.
discern	Passed to to_lodes_form() if data is in alluvia format.
negate.strata	A vector of values of the stratum aesthetic to be treated as negative (will ignore missing values with a warning).
aggregate.y	Deprecated alias for cement.alluvia.
cement.alluvia	Logical value indicating whether to aggregate y values over equivalent alluvia before computing lode and flow positions.
lode.guidance	The function to prioritize the axis variables for ordering the lodes within each stratum, or else a character string identifying the function. Character options are "zigzag", "frontback", "backfront", "forward", and "backward" (see lode-guidance-functions).
lode.ordering	**Deprecated in favor of the** order **aesthetic.** A list (of length the number of axes) of integer vectors (each of length the number of rows of data) or NULL

entries (indicating no imposed ordering), or else a numeric matrix of corresponding dimensions, giving the preferred ordering of alluvia at each axis. This will be used to order the lodes within each stratum by sorting the lodes first by stratum, then by the provided vectors, and lastly by remaining factors (if the vectors contain duplicate entries and therefore do not completely determine the lode orderings).

`aes.bind` At what grouping level, if any, to prioritize differentiation aesthetics when ordering the lodes within each stratum. Defaults to `"none"` (no aesthetic binding) with intermediate option `"flows"` to bind aesthetics after stratifying by axes linked to the index axis (the one adjacent axis in `stat_flow()`; all remaining axes in `stat_alluvium()`) and strongest option `"alluvia"` to bind aesthetics after stratifying by the index axis but before stratifying by linked axes (only available for `stat_alluvium()`). Stratification by any axis is done with respect to the strata at that axis, after separating positive and negative strata, consistent with the values of `decreasing`, `reverse`, and `absolute`. Thus, if `"none"`, then lode orderings will not depend on aesthetic variables. All aesthetic variables are used, in the order in which they are specified in `aes()`.

`infer.label` Logical; whether to assign the `stratum` or `alluvium` variable to the `label` aesthetic. Defaults to `FALSE`, and requires that no `label` aesthetic is assigned. This parameter is intended for use only with data in alluva form, which are converted to lode form before the statistical transformation. Deprecated; use `ggplot2::after_stat()` instead.

`min.y`, `max.y` Numeric; bounds on the heights of the strata to be rendered. Use these bounds to exclude strata outside a certain range, for example when labeling strata using `ggplot2::geom_text()`.

`na.rm` Logical: if `FALSE`, the default, `NA` lodes are not included; if `TRUE`, `NA` lodes constitute a separate category, plotted in grey (regardless of the color scheme).

`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()`.

Aesthetics

`stat_alluvium`, `stat_flow`, and `stat_stratum` require one of two sets of aesthetics:

- `x` and at least one of `alluvium` and `stratum`
- any number of `axis[0-9]*` (`axis1`, `axis2`, etc.)

Use `x`, `alluvium`, and/or `stratum` for data in lodes format and `axis[0-9]*` for data in alluvia format (see `alluvial-data`). Arguments to parameters inconsistent with the format will be ignored. Additionally, each `stat_*()` accepts the following optional aesthetics:

- `y`

- weight
- order
- group
- label

y controls the heights of the alluvia, and may be aggregated across equivalent observations. weight applies to the computed variables (see that section below) but does not affect the positional aesthetics. order, recognized by stat_alluvium() and stat_flow(), is used to arrange the lodes within each stratum. It tolerates duplicates and takes precedence over the differentiation aesthetics (when aes.bind is not "none") and lode guidance with respect to the remaining axes. (It replaces the deprecated parameter lode.ordering.) group is used internally; arguments are ignored. label is used to label the strata or lodes and must take a unique value across the observations within each stratum or lode.

These and any other aesthetics are aggregated as follows: Numeric aesthetics, including y, are summed. Character and factor aesthetics, including label, are assigned to strata or lodes provided they take unique values across the observations within each (and are otherwise assigned NA).

Computed variables

These can be used with ggplot2::after_stat() to control aesthetic evaluation.

n number of cases in lode

count cumulative weight of lode

prop weighted proportion of lode

stratum value of variable used to define strata

deposit order in which (signed) strata are deposited

lode lode label distilled from alluvia (stat_alluvium() and stat_flow() only)

flow direction of flow "to" or "from" from its axis (stat_flow() only)

The numerical variables n, count, and prop are calculated after the data are grouped by x and weighted by weight (in addition to y). The integer variable deposit is used internally to sort the data before calculating heights. The character variable lode is obtained from alluvium according to distill.

Package options

stat_stratum, stat_alluvium, and stat_flow order strata and lodes according to the values of several parameters, which must be held fixed across every layer in an alluvial plot. These package-specific options set global values for these parameters that will be defaulted to when not manually set:

- ggalluvial.decreasing (each stat_*): defaults to NA.
- ggalluvial.reverse (each stat_*): defaults to TRUE.
- ggalluvial.absolute (each stat_*): defaults to TRUE.
- ggalluvial.cement.alluvia (stat_alluvium): defaults to FALSE.
- ggalluvial.lode.guidance (stat_alluvium): defaults to "zigzag".
- ggalluvial.aes.bind (stat_alluvium and stat_flow): defaults to "none".

See base::options() for how to use options.

Defunct parameters

The previously defunct parameters `weight` and `aggregate.wts` have been discontinued. Use `y` and `cement.alluvia` instead.

See Also

`ggplot2::layer()` for additional arguments and `geom_alluvium()`, `geom_lode()`, and `geom_flow()` for the corresponding geoms.

Other alluvial stat layers: `stat_flow()`, `stat_stratum()`

Examples

```
# illustrate positioning
ggplot(as.data.frame(Titanic),
       aes(y = Freq,
           axis1 = Class, axis2 = Sex, axis3 = Age,
           color = Survived)) +
  stat_stratum(geom = "errorbar") +
  geom_line(stat = "alluvium") +
  stat_alluvium(geom = "pointrange") +
  geom_text(stat = "stratum", aes(label = after_stat(stratum))) +
  scale_x_discrete(limits = c("Class", "Sex", "Age"))

# lode ordering examples
gg <- ggplot(as.data.frame(Titanic),
             aes(y = Freq,
                 axis1 = Class, axis2 = Sex, axis3 = Age)) +
  geom_stratum() +
  geom_text(stat = "stratum", aes(label = after_stat(stratum))) +
  scale_x_discrete(limits = c("Class", "Sex", "Age"))
# use of lode controls
gg + geom_flow(aes(fill = Survived, alpha = Sex), stat = "alluvium",
               lode.guidance = "forward")
# prioritize aesthetic binding
gg + geom_flow(aes(fill = Survived, alpha = Sex), stat = "alluvium",
               aes.bind = "alluvia", lode.guidance = "forward")
# use of custom lode order
gg + geom_flow(aes(fill = Survived, alpha = Sex, order = sample(x = 32)),
               stat = "alluvium")
# use of custom luide guidance function
lode_custom <- function(n, i) {
  stopifnot(n == 3)
  switch(
    i,
    `1` = 1:3,
    `2` = c(2, 3, 1),
    `3` = 3:1
  )
}
gg + geom_flow(aes(fill = Survived, alpha = Sex), stat = "alluvium",
               aes.bind = "flow", lode.guidance = lode_custom)
```

```
# omit missing elements & reverse the `y` axis
ggplot(ggalluvial::majors,
       aes(x = semester, stratum = curriculum, alluvium = student, y = 1)) +
  geom_alluvium(fill = "darkgrey", na.rm = TRUE) +
  geom_stratum(aes(fill = curriculum), color = NA, na.rm = TRUE) +
  theme_bw() +
  scale_y_reverse()


# alluvium cementation examples
gg <- ggplot(ggalluvial::majors,
             aes(x = semester, stratum = curriculum, alluvium = student,
                 fill = curriculum)) +
  geom_stratum()
# diagram with outlined alluvia and labels
gg + geom_flow(stat = "alluvium", color = "black") +
  geom_text(aes(label = after_stat(lode)), stat = "alluvium")
# cemented diagram with default distillation (first most common alluvium)
gg +
  geom_flow(stat = "alluvium", color = "black", cement.alluvia = TRUE) +
  geom_text(aes(label = after_stat(lode)), stat = "alluvium",
            cement.alluvia = TRUE)
# cemented diagram with custom label distillation
gg +
  geom_flow(stat = "alluvium", color = "black", cement.alluvia = TRUE) +
  geom_text(aes(label = after_stat(lode)), stat = "alluvium",
            cement.alluvia = TRUE,
            distill = function(x) paste(x, collapse = "; "))



data(babynames, package = "babynames")
# a discontiguous alluvium
bn <- subset(babynames, prop >= .01 & sex == "F" & year > 1962 & year < 1968)
ggplot(data = bn,
       aes(x = year, alluvium = name, y = prop)) +
  geom_alluvium(aes(fill = name, color = name == "Tammy"),
                decreasing = TRUE, show.legend = FALSE) +
  scale_color_manual(values = c("#00000000", "#000000"))
# expanded to include missing values
bn2 <- merge(bn,
             expand.grid(year = unique(bn$year), name = unique(bn$name)),
             all = TRUE)
ggplot(data = bn2,
       aes(x = year, alluvium = name, y = prop)) +
  geom_alluvium(aes(fill = name, color = name == "Tammy"),
                decreasing = TRUE, show.legend = FALSE) +
  scale_color_manual(values = c("#00000000", "#000000"))
# with missing values filled in with zeros
bn2$prop[is.na(bn2$prop)] <- 0
ggplot(data = bn2,
       aes(x = year, alluvium = name, y = prop)) +
```

```
  geom_alluvium(aes(fill = name, color = name == "Tammy"),
                decreasing = TRUE, show.legend = FALSE) +
  scale_color_manual(values = c("#00000000", "#000000"))


# use negative y values to encode deaths versus survivals
titanic <- as.data.frame(Titanic)
titanic <- transform(titanic, Lives = Freq * (-1) ^ (Survived == "No"))
ggplot(subset(titanic, Class != "Crew"),
       aes(axis1 = Class, axis2 = Sex, axis3 = Age, y = Lives)) +
  geom_alluvium(aes(alpha = Survived, fill = Class), absolute = FALSE) +
  geom_stratum(absolute = FALSE) +
  geom_text(stat = "stratum", aes(label = after_stat(stratum)),
            absolute = FALSE) +
  scale_x_discrete(limits = c("Class", "Sex", "Age"), expand = c(.1, .05)) +
  scale_alpha_discrete(range = c(.25, .75), guide = "none")

# faceting with common alluvia
ggplot(titanic, aes(y = Freq, axis1 = Class, axis2 = Sex, axis3 = Age)) +
  facet_wrap(~ Survived) +
  geom_alluvium() +
  geom_stratum() +
  geom_text(stat = "stratum", aes(label = after_stat(stratum)))
ggplot(transform(alluvial::Refugees, id = 1),
       aes(y = refugees, x = year, alluvium = id)) +
  facet_wrap(~ country) +
  geom_alluvium(alpha = .75, color = "darkgrey") +
  scale_x_continuous(breaks = seq(2004, 2012, 4))
```

`stat_flow` *Flow positions*

Description

Given a dataset with alluvial structure, `stat_flow` calculates the centroids (`x` and `y`) and heights (`ymin` and `ymax`) of the flows between each pair of adjacent axes.

Usage

```
stat_flow(
  mapping = NULL,
  data = NULL,
  geom = "flow",
  position = "identity",
  decreasing = NULL,
  reverse = NULL,
  absolute = NULL,
  discern = FALSE,
  negate.strata = NULL,
```

```
  aes.bind = NULL,
  infer.label = FALSE,
  min.y = NULL,
  max.y = NULL,
  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)).
geom	The geometric object to use display the data; override the default.
position	Position adjustment, either as a string naming the adjustment (e.g. "jitter" to use position_jitter), or the result of a call to a position adjustment function. Use the latter if you need to change the settings of the adjustment.
decreasing	Logical; whether to arrange the strata at each axis in the order of the variable values (NA, the default), in ascending order of totals (largest on top, FALSE), or in descending order of totals (largest on bottom, TRUE).
reverse	Logical; if decreasing is NA, whether to arrange the strata at each axis in the reverse order of the variable values, so that they match the order of the values in the legend. Ignored if decreasing is not NA. Defaults to TRUE.
absolute	Logical; if some cases or strata are negative, whether to arrange them (respecting decreasing and reverse) using negative or absolute values of y.
discern	Passed to to_lodes_form() if data is in alluvia format.
negate.strata	A vector of values of the stratum aesthetic to be treated as negative (will ignore missing values with a warning).
aes.bind	At what grouping level, if any, to prioritize differentiation aesthetics when ordering the lodes within each stratum. Defaults to "none" (no aesthetic binding) with intermediate option "flows" to bind aesthetics after stratifying by axes linked to the index axis (the one adjacent axis in stat_flow(); all remaining axes in stat_alluvium()) and strongest option "alluvia" to bind aesthetics after stratifying by the index axis but before stratifying by linked axes (only available for stat_alluvium()). Stratification by any axis is done with respect

to the strata at that axis, after separating positive and negative strata, consistent with the values of `decreasing`, `reverse`, and `absolute`. Thus, if `"none"`, then lode orderings will not depend on aesthetic variables. All aesthetic variables are used, in the order in which they are specified in `aes()`.

`infer.label` Logical; whether to assign the `stratum` or `alluvium` variable to the `label` aesthetic. Defaults to `FALSE`, and requires that no `label` aesthetic is assigned. This parameter is intended for use only with data in alluva form, which are converted to lode form before the statistical transformation. Deprecated; use `ggplot2::after_stat()` instead.

`min.y`, `max.y` Numeric; bounds on the heights of the strata to be rendered. Use these bounds to exclude strata outside a certain range, for example when labeling strata using `ggplot2::geom_text()`.

`na.rm` Logical: if `FALSE`, the default, `NA` lodes are not included; if `TRUE`, `NA` lodes constitute a separate category, plotted in grey (regardless of the color scheme).

`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()`.

Aesthetics

`stat_alluvium`, `stat_flow`, and `stat_stratum` require one of two sets of aesthetics:

- `x` and at least one of `alluvium` and `stratum`
- any number of `axis[0-9]*` (`axis1`, `axis2`, etc.)

Use `x`, `alluvium`, and/or `stratum` for data in lodes format and `axis[0-9]*` for data in alluvia format (see `alluvial-data`). Arguments to parameters inconsistent with the format will be ignored. Additionally, each `stat_*()` accepts the following optional aesthetics:

- `y`
- `weight`
- `order`
- `group`
- `label`

`y` controls the heights of the alluvia, and may be aggregated across equivalent observations. `weight` applies to the computed variables (see that section below) but does not affect the positional aesthetics. `order`, recognized by `stat_alluvium()` and `stat_flow()`, is used to arrange the lodes within each stratum. It tolerates duplicates and takes precedence over the differentiation aesthetics (when `aes.bind` is not `"none"`) and lode guidance with respect to the remaining axes. (It replaces the deprecated parameter `lode.ordering`.) `group` is used internally; arguments are ignored. `label` is used to label the strata or lodes and must take a unique value across the observations within each stratum or lode.

These and any other aesthetics are aggregated as follows: Numeric aesthetics, including y, are summed. Character and factor aesthetics, including label, are assigned to strata or lodes provided they take unique values across the observations within each (and are otherwise assigned NA).

Computed variables

These can be used with ggplot2::after_stat() to control aesthetic evaluation.

n number of cases in lode

count cumulative weight of lode

prop weighted proportion of lode

stratum value of variable used to define strata

deposit order in which (signed) strata are deposited

lode lode label distilled from alluvia (stat_alluvium() and stat_flow() only)

flow direction of flow "to" or "from" from its axis (stat_flow() only)

The numerical variables n, count, and prop are calculated after the data are grouped by x and weighted by weight (in addition to y). The integer variable deposit is used internally to sort the data before calculating heights. The character variable lode is obtained from alluvium according to distill.

Package options

stat_stratum, stat_alluvium, and stat_flow order strata and lodes according to the values of several parameters, which must be held fixed across every layer in an alluvial plot. These package-specific options set global values for these parameters that will be defaulted to when not manually set:

- ggalluvial.decreasing (each stat_*): defaults to NA.
- ggalluvial.reverse (each stat_*): defaults to TRUE.
- ggalluvial.absolute (each stat_*): defaults to TRUE.
- ggalluvial.cement.alluvia (stat_alluvium): defaults to FALSE.
- ggalluvial.lode.guidance (stat_alluvium): defaults to "zigzag".
- ggalluvial.aes.bind (stat_alluvium and stat_flow): defaults to "none".

See base::options() for how to use options.

Defunct parameters

The previously defunct parameters weight and aggregate.wts have been discontinued. Use y and cement.alluvia instead.

See Also

ggplot2::layer() for additional arguments and geom_alluvium() and geom_flow() for the corresponding geoms.

Other alluvial stat layers: stat_alluvium(), stat_stratum()

Examples

```
# illustrate positioning
ggplot(as.data.frame(Titanic),
       aes(y = Freq,
           axis1 = Class, axis2 = Sex, axis3 = Age,
           color = Survived)) +
  stat_stratum(geom = "errorbar") +
  geom_line(stat = "flow") +
  stat_flow(geom = "pointrange") +
  geom_text(stat = "stratum", aes(label = after_stat(stratum))) +
  scale_x_discrete(limits = c("Class", "Sex", "Age"))

# alluvium--flow comparison
data(vaccinations)
gg <- ggplot(vaccinations,
             aes(x = survey, stratum = response, alluvium = subject,
                 y = freq, fill = response)) +
  geom_stratum(alpha = .5) +
  geom_text(aes(label = response), stat = "stratum")
# rightward alluvial aesthetics for vaccine survey data
gg + geom_flow(stat = "alluvium", lode.guidance = "forward")
# memoryless flows for vaccine survey data
gg + geom_flow()

# size filter examples
gg <- ggplot(vaccinations,
       aes(y = freq,
           x = survey, stratum = response, alluvium = subject,
           fill = response, label = response)) +
  stat_stratum(alpha = .5) +
  geom_text(stat = "stratum")
# omit small flows
gg + geom_flow(min.y = 50)
# omit large flows
gg + geom_flow(max.y = 100)

# negate missing entries
ggplot(vaccinations,
       aes(y = freq,
           x = survey, stratum = response, alluvium = subject,
           fill = response, label = response,
           alpha = response != "Missing")) +
  stat_stratum(negate.strata = "Missing") +
  geom_flow(negate.strata = "Missing") +
  geom_text(stat = "stratum", alpha = 1, negate.strata = "Missing") +
  scale_alpha_discrete(range = c(.2, .6)) +
  guides(alpha = "none")


# aesthetics that vary betwween and within strata
data(vaccinations)
vaccinations$subgroup <- LETTERS[1:2][rbinom(
```

```
  n = length(unique(vaccinations$subject)), size = 1, prob = .5
) + 1][vaccinations$subject]
ggplot(vaccinations,
       aes(x = survey, stratum = response, alluvium = subject,
           y = freq, fill = response, label = response)) +
  geom_flow(aes(alpha = subgroup)) +
  scale_alpha_discrete(range = c(1/3, 2/3)) +
  geom_stratum(alpha = .5) +
  geom_text(stat = "stratum")
# can even set aesthetics that vary both ways
ggplot(vaccinations,
       aes(x = survey, stratum = response, alluvium = subject,
           y = freq, label = response)) +
  geom_flow(aes(fill = interaction(response, subgroup)), aes.bind = "flows") +
  scale_alpha_discrete(range = c(1/3, 2/3)) +
  geom_stratum(alpha = .5) +
  geom_text(stat = "stratum")
```

`stat_stratum` *Stratum positions*

Description

Given a dataset with alluvial structure, `stat_stratum` calculates the centroids (`x` and `y`) and heights (`ymin` and `ymax`) of the strata at each axis.

Usage

```
stat_stratum(
  mapping = NULL,
  data = NULL,
  geom = "stratum",
  position = "identity",
  decreasing = NULL,
  reverse = NULL,
  absolute = NULL,
  discern = FALSE,
  distill = "first",
  negate.strata = NULL,
  infer.label = FALSE,
  label.strata = NULL,
  min.y = NULL,
  max.y = NULL,
  min.height = NULL,
  max.height = NULL,
  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)).
geom	The geometric object to use display the data; override the default.
position	Position adjustment, either as a string naming the adjustment (e.g. "jitter" to use position_jitter), or the result of a call to a position adjustment function. Use the latter if you need to change the settings of the adjustment.
decreasing	Logical; whether to arrange the strata at each axis in the order of the variable values (NA, the default), in ascending order of totals (largest on top, FALSE), or in descending order of totals (largest on bottom, TRUE).
reverse	Logical; if decreasing is NA, whether to arrange the strata at each axis in the reverse order of the variable values, so that they match the order of the values in the legend. Ignored if decreasing is not NA. Defaults to TRUE.
absolute	Logical; if some cases or strata are negative, whether to arrange them (respecting decreasing and reverse) using negative or absolute values of y.
discern	Passed to to_lodes_form() if data is in alluvia format.
distill	A function (or its name) to be used to distill alluvium values to a single lode label, accessible via ggplot2::after_stat() (similar to its behavior in to_alluvia_form()). It recognizes three character values: "first" (the default) and "last" as defined in **dplyr**; and "most" (which returns the first modal value).
negate.strata	A vector of values of the stratum aesthetic to be treated as negative (will ignore missing values with a warning).
infer.label	Logical; whether to assign the stratum or alluvium variable to the label aesthetic. Defaults to FALSE, and requires that no label aesthetic is assigned. This parameter is intended for use only with data in alluva form, which are converted to lode form before the statistical transformation. Deprecated; use ggplot2::after_stat() instead.
label.strata	Defunct; alias for infer.label.
min.y, max.y	Numeric; bounds on the heights of the strata to be rendered. Use these bounds to exclude strata outside a certain range, for example when labeling strata using ggplot2::geom_text().

`min.height`, `max.height`
: Deprecated aliases for `min.y` and `max.y`.

`na.rm`
: Logical: if `FALSE`, the default, `NA` lodes are not included; if `TRUE`, `NA` lodes constitute a separate category, plotted in grey (regardless of the color scheme).

`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()`.

Aesthetics

`stat_alluvium`, `stat_flow`, and `stat_stratum` require one of two sets of aesthetics:

- `x` and at least one of `alluvium` and `stratum`
- any number of `axis[0-9]*` (`axis1`, `axis2`, etc.)

Use `x`, `alluvium`, and/or `stratum` for data in lodes format and `axis[0-9]*` for data in alluvia format (see `alluvial-data`). Arguments to parameters inconsistent with the format will be ignored. Additionally, each `stat_*()` accepts the following optional aesthetics:

- `y`
- `weight`
- `order`
- `group`
- `label`

`y` controls the heights of the alluvia, and may be aggregated across equivalent observations. `weight` applies to the computed variables (see that section below) but does not affect the positional aesthetics. `order`, recognized by `stat_alluvium()` and `stat_flow()`, is used to arrange the lodes within each stratum. It tolerates duplicates and takes precedence over the differentiation aesthetics (when `aes.bind` is not `"none"`) and lode guidance with respect to the remaining axes. (It replaces the deprecated parameter `lode.ordering`.) `group` is used internally; arguments are ignored. `label` is used to label the strata or lodes and must take a unique value across the observations within each stratum or lode.

These and any other aesthetics are aggregated as follows: Numeric aesthetics, including `y`, are summed. Character and factor aesthetics, including `label`, are assigned to strata or lodes provided they take unique values across the observations within each (and are otherwise assigned `NA`).

Computed variables

These can be used with `ggplot2::after_stat()` to control aesthetic evaluation.

`n` number of cases in lode

`count` cumulative weight of lode

`prop` weighted proportion of lode

`stratum` value of variable used to define strata

`deposit` order in which (signed) strata are deposited

`lode` lode label distilled from alluvia (`stat_alluvium()` and `stat_flow()` only)

`flow` direction of flow `"to"` or `"from"` from its axis (`stat_flow()` only)

The numerical variables `n`, `count`, and `prop` are calculated after the data are grouped by `x` and weighted by `weight` (in addition to `y`). The integer variable `deposit` is used internally to sort the data before calculating heights. The character variable `lode` is obtained from `alluvium` according to `distill`.

Package options

`stat_stratum`, `stat_alluvium`, and `stat_flow` order strata and lodes according to the values of several parameters, which must be held fixed across every layer in an alluvial plot. These package-specific options set global values for these parameters that will be defaulted to when not manually set:

- `ggalluvial.decreasing` (each `stat_*`): defaults to `NA`.
- `ggalluvial.reverse` (each `stat_*`): defaults to `TRUE`.
- `ggalluvial.absolute` (each `stat_*`): defaults to `TRUE`.
- `ggalluvial.cement.alluvia` (`stat_alluvium`): defaults to `FALSE`.
- `ggalluvial.lode.guidance` (`stat_alluvium`): defaults to `"zigzag"`.
- `ggalluvial.aes.bind` (`stat_alluvium` and `stat_flow`): defaults to `"none"`.

See `base::options()` for how to use options.

Defunct parameters

The previously defunct parameters `weight` and `aggregate.wts` have been discontinued. Use `y` and `cement.alluvia` instead.

See Also

`ggplot2::layer()` for additional arguments and `geom_stratum()` for the corresponding geom.

Other alluvial stat layers: `stat_alluvium()`, `stat_flow()`

Examples

```
data(vaccinations)
# only `stratum` assignment is necessary to generate strata
ggplot(vaccinations,
       aes(y = freq,
           x = survey, stratum = response,
           fill = response)) +
  stat_stratum(width = .5)

# lode data, positioning with y labels
```

```
ggplot(vaccinations,
       aes(y = freq,
           x = survey, stratum = response, alluvium = subject,
           label = after_stat(count))) +
  stat_stratum(geom = "errorbar") +
  geom_text(stat = "stratum")
# alluvium data, positioning with stratum labels
ggplot(as.data.frame(Titanic),
       aes(y = Freq,
           axis1 = Class, axis2 = Sex, axis3 = Age, axis4 = Survived)) +
  geom_text(stat = "stratum", aes(label = after_stat(stratum))) +
  stat_stratum(geom = "errorbar") +
  scale_x_discrete(limits = c("Class", "Sex", "Age", "Survived"))

# omit labels for strata outside a y range
ggplot(vaccinations,
       aes(y = freq,
           x = survey, stratum = response,
           fill = response, label = response)) +
  stat_stratum(width = .5) +
  geom_text(stat = "stratum", min.y = 100)

# date-valued axis variables
ggplot(vaccinations,
       aes(x = end_date, y = freq, stratum = response, alluvium = subject,
           fill = response)) +
  stat_alluvium(geom = "flow", lode.guidance = "forward",
                width = 30) +
  stat_stratum(width = 30) +
  labs(x = "Survey date", y = "Number of respondents")

admissions <- as.data.frame(UCBAdmissions)
admissions <- transform(admissions, Count = Freq * (-1) ^ (Admit == "Rejected"))
# use negative y values to encode rejection versus acceptance
ggplot(admissions,
       aes(y = Count, axis1 = Dept, axis2 = Gender)) +
  geom_alluvium(aes(fill = Dept), width = 1/12) +
  geom_stratum(width = 1/12, fill = "black", color = "grey") +
  geom_label(stat = "stratum", aes(label = after_stat(stratum)), min.y = 200) +
  scale_x_discrete(limits = c("Department", "Gender"), expand = c(.05, .05))
# computed variable 'deposit' indicates order of each signed stratum
ggplot(admissions,
       aes(y = Count, axis1 = Dept, axis2 = Gender)) +
  geom_alluvium(aes(fill = Dept), width = 1/12) +
  geom_stratum(width = 1/12, fill = "black", color = "grey") +
  geom_text(stat = "stratum", aes(label = after_stat(deposit)),
            color = "white") +
  scale_x_discrete(limits = c("Department", "Gender"), expand = c(.05, .05))
# fixed-width strata with acceptance and rejection totals
ggplot(admissions,
       aes(y = sign(Count), weight = Count, axis1 = Dept, axis2 = Gender)) +
  geom_alluvium(aes(fill = Dept), width = 1/8) +
  geom_stratum(width = 1/8, fill = "black", color = "grey") +
```

```
  geom_text(stat = "stratum",
            aes(label = paste0(stratum,
                               ifelse(nchar(as.character(stratum)) == 1L,
                                      ": ", "\n"),
                               after_stat(n))),
            color = "white", size = 3) +
  scale_x_discrete(limits = c("Department", "Gender"), expand = c(.05, .05))
```

`vaccinations` *Influenza vaccination survey responses*

Description

This data set is aggregated from three RAND American Life Panel (ALP) surveys that asked respondents their probability of vaccinating for influenza. Their responses were discretized to "Never" (0%), "Always" (100%), or "Sometimes" (any other value). After merging, missing responses were coded as "Missing" and respondents were grouped and counted by all three coded responses. The pre-processed data were kindly contributed by Raffaele Vardavas, and the complete surveys are freely available at the ALP website.

Usage

```
vaccinations
```

Format

A data frame with 117 rows and 5 variables:

`freq` number of respondents represented in each row

`subject` identifier linking respondents across surveys

`survey` survey designation from the ALP website

`start_date` start date of survey

`end_date` end date of survey

`response` discretized probability of vaccinating for influenza

Source

https://alpdata.rand.org/

Index