Package: ggsankeyfier (via r-universe)

August 30, 2024

Type Package

Title Create Sankey and Alluvial Diagrams Using 'ggplot2'

Version 0.1.8.0005

Author Pepijn de Vries [aut, cre, dtc] (0000-0002-7961-6646), Gerjan Piet [dtc] (0000-0003-0702-1624), Ruud Jongbloed [dtc] (0000-0002-9378-5382), Anne Grundlehner [dtc] (0000-0003-3375-3511), Jacqueline Tamis [dtc] (0000-0002-8206-5830)

Maintainer Pepijn de Vries <pepijn.devries@outlook.com>

Description Sankey and alluvial diagrams visualise flows of quantities across stages in stacked bars. This package makes it easy to create such diagrams using 'ggplot2'.

Depends R (>= 4.1.0)

Imports dplyr, ggplot2 (>= 3.5.0), grid (>= 4.1.0), gridBezier, methods, rlang, scales, stats, tidyr, vwline

Suggests knitr, rmarkdown, stringr, svglite, testthat (>= 3.0.0), vdiffr

License GPL (>= 3)

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

Roxygen list(markdown = TRUE)

Collate 'draw_edges.r' 'draw_key.r' 'ecosystem_services.r' 'geom_segment.r' 'scale_waist.r' 'geom_edge.r' 'position_helpers.r' 'geom_node.r' 'ggsankeyfier-package.r' 'imports.r' 'pivot_stages.r' 'position.r' 'short_cuts.r' 'stat_edge.r' 'stat_node_helpers.r' 'stat_node.r'

VignetteBuilder knitr

Config/testthat/edition 3

URL https://pepijn-devries.github.io/ggsankeyfier/, https://github.com/pepijn-devries/ggsankeyfier/

BugReports https://github.com/pepijn-devries/ggsankeyfier/issues

Repository https://pepijn-devries.r-universe.dev

RemoteUrl https://github.com/pepijn-devries/ggsankeyfier

RemoteRef master

RemoteSha c7c6204dfc7e9a7eebfa644bde7d96d5b9bf440b

Contents

`draw_key_sankeyedge` *Key glyphs for legends*

Description

Each geom has an associated function that draws the key when the geom needs to be displayed in a legend. These functions are called `draw_key_*()`, where `*` stands for the name of the respective key glyph. The key glyphs can be customized for individual geoms by providing a geom with the `key_glyph` argument (see `layer()` or examples below.)

Usage

```
draw_key_sankeyedge(data, params, size)

draw_key_sankeynode(data, params, size)
```

Arguments

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

Value

A grid grob.

Author(s)

Pepijn de Vries

Examples

```
## The key glyph for sankey diagrams can be applied to different geoms as well.
## In the example below it is applied to a histogram

library(ggplot2)
ggplot(data.frame(x = rnorm(100), groups = rep(letters[1:2], 2)),
       aes(x = x, fill = groups)) +
  geom_histogram(key_glyph = draw_key_sankeyedge, binwidth = 0.2, alpha = 1)
ggplot(data.frame(x = rnorm(100), groups = rep(letters[1:2], 2)),
       aes(x = x, fill = groups)) +
  geom_histogram(key_glyph = draw_key_sankeynode, binwidth = 0.2)
```

---

ecosystem_services *Data on risks to supply ecosystem services*

Description

Data indicating a risk resulting from anthropological activities to the marine ecosystem and its capacity to supply services. This data set serves (aggregated from Piet *et al.* (submitted)) as an example to illustrate the package's features.

Format

`ecosystem_services` is a `data.frame` with 3421 rows and 8 columns. The columns are:

- `activity_type`: Type of activities that pose a risk
- `activity_realm`: Aggregation of activity types
- `pressure_cat`: Category of pressures exerted by the activities and eventually pose a risk to the ecosystem.
- `biotic_group`: Biotic groups affected by the pressures.
- `biotic_realm`: Aggregation of biotic groups
- `service_division`: Division of ecosystem services that are provided by the biotic groups and affected by the activities.
- `service_section`: Aggregation of service divisions.
- `RCSES`: 'Risk to Capacity to Supply Ecosystem Services'. A numerical score reflecting the amount of risk for the ecosystem to supply specific services. For more details see Piet et al. (submitted)

This data.frame is in a wide oriented format, typical for most common applications. Each row in the data.frame represents a unique pathway where each activity_tpe poses a risk to an ecosystem sevice_division, via a pressure_cat and biotic_group. Each column either contains information on a specific stage or the overall quantifier (in this case RCSES).

In its present form it is not suitable to directly plot as a Sankey diagram. For that purpose it needs to be pivoted with pivot_stages_longer(). Two different variants are prepared with this function: ecosystem_services_pivot1 and ecosystem_services_pivot2.

The latter pivot contains service_section as an extra feature which can be used for additional decoration of a Sankey diagram. It is therefore more detailed than the first alternative.

ecosystem_services_pivot1 is a data.frame with 112 rows and 5 columns. Columns are:

- RCSES: See above at ecosystem_services.
- edge_id: Unique numerical identifier for each edge in a Sankey diagram.
- connector: One of 'from' or 'to', indicating whether we are looking at the start or the end of an edge.
- node: A collection of activity_realm, pressure_cat, biotic_realm and service_section values from the ecosystem_services data.frame.
- stage: Stages in a Sankey diagram formed by the columns activity_realm, pressure_cat, biotic_realm and service_section from the ecosystem_services data.frame.

ecosystem_services_pivot1 is created from ecosystem_services using pivot_stages_longer() and can be used directly in a Sankey diagram (using geom_sankeynode() and geom_sankeyedge())

ecosystem_services_pivot2 is a data.frame with 252 rows and 6. It is the same as ecosystem_services_pivot1 with the exception of a distinct extra column service_division which allows for more detailed aesthetics in a Sankey diagram.

Author(s)

Pepijn de Vries, Gerjan Piet, Jacob Bentley, Ruud Jongbloed, Anne Grundlehner, Jacqueline Tamis

References

Piet GJ, Bentley J, Jongbloed RH, Grundlehner A, Tamis JE, De Vries P (submitted) A Cumulative Impact Assessment on the North Sea Capacity to Supply Ecosystem Services. doi:10.2139/ssrn.4450241

Examples

```
data("ecosystem_services")
library(ggplot2)

if (requireNamespace("stringr")) {
  library(stringr)

  pos <- position_sankey(v_space = "auto", align = "justify")
  pos_text <- position_sankey(v_space = "auto", align = "justify", nudge_x = 0.1)

  ## A simplified version of the Sankey diagram as published by Piet _et al._ (submitted)
```

```
  ggplot(ecosystem_services |>
           pivot_stages_longer(
             c("activity_type", "pressure_cat", "biotic_group", "service_section"),
             "RCSES"),
         aes(x = stage, y = RCSES, group = node, connector = connector, edge_id = edge_id)) +
    geom_sankeyedge(aes(fill = RCSES), position = pos) +
    geom_sankeynode(position = pos) +
    geom_text(aes(label = str_wrap(node, 20)), position = pos_text, stat = "sankeynode",
              hjust = 0, cex = 2) +
    scale_fill_viridis_c(option = "turbo", trans = "sqrt") +
    theme_minimal()
}
```

GeomSankeynode *Bars representing nodes in a Sankey diagram*

Description

In a Sankey diagram nodes are depicted as stacked bars, possibly with vertical spacing between them. Use `geom_sankeynode()` to add nodes to your Sankey diagram. If you combine the nodes with `geom_sankeyedge()`, make sure that both use the same `position` object.

Usage

```
GeomSankeynode

geom_sankeynode(
  mapping = NULL,
  data = NULL,
  stat = "sankeynode",
  position = "sankey",
  na.rm = FALSE,
  show.legend = NA,
  width = "auto",
  align = c("bottom", "top", "center", "justify"),
  order = c("ascending", "descending", "as_is"),
  h_space = "auto",
  v_space = 0,
  nudge_x = 0,
  nudge_y = 0,
  split_nodes = FALSE,
  split_tol = 0.001,
  direction = c("forward", "backward"),
  inherit.aes = TRUE,
  ...
)
```

Arguments

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

`order` A `character` indicating the method to be used for the order of stacking nodes and edges in a plot. Should be one of: `ascending` (default), sorts nodes and edges from large to small (largest on top); `descending` sorts nodes and edges from small to large (smallest on top); `as_is` will leave the order of nodes and edges as they are in `data`.

`h_space` Horizontal space between split nodes (`numeric`). This argument is ignored when `split_nodes == FALSE`. Use `"auto"` to automatically position split nodes.

`v_space` Vertical space between nodes (`numeric`). When set to zero (`0`), the Sankey diagram becomes an alluvial plot. Use `"auto"` to automatically determine a suitable vertical space.

`nudge_x, nudge_y` Horizontal and vertical adjustment to nudge items by. Can be useful for offsetting labels.

`split_nodes` A `logical` value indicating whether the source and destination nodes should be depicted as separate boxes.

`split_tol` When the relative node size (resulting source and destination edges) differs more than this fraction, the node will be displayed as two separate bars.

`direction` One of `"forward"` (default) or `"backward"`. When set to `"backward"` the direction of the edges will be inverted. In most cases this parameter won't affect the plot. It can be helpful when you want to decorate the end of an edge (instead of the start; see examples).

`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()`.

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

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

Format

An object of class GeomSankeynode (inherits from GeomBar, GeomRect, Geom, ggproto, gg) of length 6.

Details

This ggplot2 layer depicts the size of all connected edges as a bar. The height of of each bar is determined by the sum of y aesthetic in each group. When the sum of edges that flow to a bar differ more than split_tol compared to the edges that flow from the same node, a vertical split is introduced in the node.

Value

Returns a ggplot2::layer() which can be added to a ggplot2::ggplot()

Aesthetics

geom_sankeynode() understands the following aesthetics (required aesthetics are in bold)

- x: Works for variables on a discrete scale. Might work for continuous variables but is not guaranteed. This variable is used to distinguish between stages in the Sankey diagram on the x axis.
- y: A continuous variable representing the width of the edges in a Sankey diagram.
- group: A discrete variable used for grouping edges to nodes in each stage. Essentially it is an identifier for the nodes.
- connector: Indicates which side of an edge is reflected by the corresponding record. Should be one of "from" or "to".
- edge_id: A unique identifier value for each edge. This identifier is used to link specific "from" and "to" records in an edge (flow).
- fill: see vignette("ggplot2-specs", "ggplot2")
- colour: see vignette("ggplot2-specs", "ggplot2")
- linetype: see vignette("ggplot2-specs", "ggplot2")
- linewidth: see vignette("ggplot2-specs", "ggplot2")
- alpha: A variable to control the opacity of an element.

Author(s)

Pepijn de Vries

Examples

```
library(ggplot2)
data("ecosystem_services")

ggplot(ecosystem_services_pivot1, aes(x = stage, y = RCSES, group = node,
                    connector = connector, edge_id = edge_id,
                    fill = node)) +
```

```
  geom_sankeynode(v_space = "auto") +
  geom_sankeyedge(v_space = "auto")
```

GeomSankeysegment *Sankey edges (flows)*

Description

geom_sankeysegment() draws a straight line between two connected nodes, geom_sankeyedge() draws a ribbon between nodes following a Bezier curved path. If you combine the edges with geom_sankeynode(), make sure that both use the same position object.

Usage

```
GeomSankeysegment

geom_sankeysegment(
  mapping = NULL,
  data = NULL,
  stat = "sankeyedge",
  position = "sankey",
  na.rm = FALSE,
  show.legend = NA,
  order = c("ascending", "descending", "as_is"),
  width = "auto",
  align = c("bottom", "top", "center", "justify"),
  h_space = "auto",
  v_space = 0,
  nudge_x = 0,
  nudge_y = 0,
  split_nodes = FALSE,
  split_tol = 0.001,
  direction = c("forward", "backward"),
  inherit.aes = TRUE,
  ...
)

GeomSankeyedge

geom_sankeyedge(
  mapping = NULL,
  data = NULL,
  stat = "sankeyedge",
  position = "sankey",
  na.rm = FALSE,
  show.legend = NA,
  slope = 0.5,
```

```
  ncp = 100,
  width = "auto",
  align = c("bottom", "top", "center", "justify"),
  order = c("ascending", "descending", "as_is"),
  h_space = "auto",
  v_space = 0,
  nudge_x = 0,
  nudge_y = 0,
  split_nodes = FALSE,
  split_tol = 0.001,
  direction = c("forward", "backward"),
  inherit.aes = TRUE,
  ...
)
```

Arguments

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

- For more information and other ways to specify the position, see the layer position documentation.

`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.

`order` A `character` indicating the method to be used for the order of stacking nodes and edges in a plot. Should be one of: `ascending` (default), sorts nodes and edges from large to small (largest on top); `descending` sorts nodes and edges from small to large (smallest on top); `as_is` will leave the order of nodes and edges as they are in `data`.

`width` Width of the node (`numeric`). When `split_nodes` is set to `TRUE` each part of the split node will have half this width. Use `"auto"` to automatically determine a suitable width.

`align` A `character` that indicates how the nodes across the stages are aligned. It can be any of `"top"`, `"bottom"`, `"center"` or `"justify"`.

`h_space` Horizontal space between split nodes (`numeric`). This argument is ignored when `split_nodes == FALSE`. Use `"auto"` to automatically position split nodes.

`v_space` Vertical space between nodes (`numeric`). When set to zero (`0`), the Sankey diagram becomes an alluvial plot. Use `"auto"` to automatically determine a suitable vertical space.

`nudge_x, nudge_y`
Horizontal and vertical adjustment to nudge items by. Can be useful for offsetting labels.

`split_nodes` A `logical` value indicating whether the source and destination nodes should be depicted as separate boxes.

`split_tol` When the relative node size (resulting source and destination edges) differs more than this fraction, the node will be displayed as two separate bars.

`direction` One of `"forward"` (default) or `"backward"`. When set to `"backward"` the direction of the edges will be inverted. In most cases this parameter won't affect the plot. It can be helpful when you want to decorate the end of an edge (instead of the start; see examples).

`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()`.

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

- Static aesthetics that are not mapped to a scale, but are at a fixed value and apply to the layer as a whole. For example, `colour = "red"` or `linewidth = 3`. The geom's documentation has an **Aesthetics** section that lists the available options. The 'required' aesthetics cannot be passed on to the

params. Please note that while passing unmapped aesthetics as vectors is technically possible, the order and required length is not guaranteed to be parallel to the input data.

- When constructing a layer using a `stat_*()` function, the `...` argument can be used to pass on parameters to the geom part of the layer. An example of this is `stat_density(geom = "area", outline.type = "both")`. The geom's documentation lists which parameters it can accept.
- Inversely, when constructing a layer using a `geom_*()` function, the `...` argument can be used to pass on parameters to the `stat` part of the layer. An example of this is `geom_area(stat = "density", adjust = 0.5)`. The stat's documentation lists which parameters it can accept.
- The `key_glyph` argument of `layer()` may also be passed on through `....` This can be one of the functions described as key glyphs, to change the display of the layer in the legend.

`slope` Slope parameter (`numeric`) for the Bezier curves used to depict the edges. Any value between 0 and 1 will work nicely. Other non-zero values will also work.

`ncp` Number of control points on the Bezier curve that forms the edge. Larger numbers will result in smoother curves, but cost more computational time. Default is 100.

Format

An object of class `GeomSankeysegment` (inherits from `GeomSegment`, `Geom`, `ggproto`, `gg`) of length 4.

An object of class `GeomSankeyedge` (inherits from `GeomSankeysegment`, `GeomSegment`, `Geom`, `ggproto`, `gg`) of length 7.

Details

This `ggplot2` layer connects between paired nodes via a Bezier curve. The width of the curve is determined by its `y` aesthetic. It will be attempted to keep the width of the curve constant along its curved path, for the targeted graphics device. When the aspect ratio of the graphics device is altered after the plot is generated, the aspect ratio maybe off. In that case render the plot again.

Value

Returns a `ggplot2::layer()` which can be added to a `ggplot2::ggplot()`

Aesthetics

`geom_sankeysegment()` and `geom_sankeyedge()` understand the following aesthetics (required aesthetics are in bold)

- `x`: Works for variables on a discrete scale. Might work for continuous variables but is not guaranteed. This variable is used to distinguish between stages in the Sankey diagram on the x axis.
- `y`: A continuous variable representing the width of the edges in a Sankey diagram.

- group: A discrete variable used for grouping edges to nodes in each stage. Essentially it is an identifier for the nodes.
- connector: Indicates which side of an edge is reflected by the corresponding record. Should be one of "from" or "to".
- edge_id: A unique identifier value for each edge. This identifier is used to link specific "from" and "to" records in an edge (flow).
- fill: see vignette("ggplot2-specs", "ggplot2")
- colour: see vignette("ggplot2-specs", "ggplot2")
- linetype: see vignette("ggplot2-specs", "ggplot2")
- linewidth: see vignette("ggplot2-specs", "ggplot2")
- alpha: A variable to control the opacity of an element.
- waist: A variable to control the width of an edge in between two nodes. Small values will create a hour glass shape, whereas large values will produce an apple shape.

Author(s)

Pepijn de Vries

Examples

```
library(ggplot2)
data("ecosystem_services")

ggplot(ecosystem_services_pivot1, aes(x = stage, y = RCSES, group = node,
                    connector = connector, edge_id = edge_id,
                    fill = node)) +
  geom_sankeynode(v_space = "auto") +
  geom_sankeyedge(v_space = "auto")
```

---

pivot_stages_longer *Pivot wide data to long for plotting as Sankey*

Description

Pivot data from a wide to a long format suitable for plotting Sankey diagrams.

Usage

```
pivot_stages_longer(
  data,
  stages_from,
  values_from,
  additional_aes_from,
  invert_nodes = FALSE
)
```

Arguments

data	A data.frame (or an object inheriting the data.frame class), which needs to be pivoted.
stages_from	A vector of column names, which represent the stages.
values_from	A vector of column names, which contains numeric values that represent the size of the edges in Sankey diagrams. When there are multiple values for a single edge, they are summed.
additional_aes_from	A vector of column names of data that you want to use to decorate elements in your Sankey diagram. This argument is optional. See also vignette("data_management") and vignette("decorating").
invert_nodes	When pivoting information from stages_from, its data is converted into a factor. Set invert_nodes to TRUE if you want to invert the order of the levels of the factor.

Details

Typically, data to be displayed as a Sankey, is collected and stored in a wide format, where each stage (i.e., x-axis of a Sankey diagram) is in a column. The ggplot2 philosophy requires the data to be in a long format, such that diagram decorations (aesthetics) can be mapped to specific columns.

This function pivots wide data in an appropriate long format, by indicating which columns contain the stages, and in which order they should appear in the Sankey.

For more details see vignette("data_management")

Value

Returns a dplyr::tibble with all the selected columns from data pivoted. The stages will be listed in the column named stage and nodes in the column named node. The result will contain two new columns: a column named connector indicating whether the row in the tibble reflects the source of an edge (value 'from') or destination of an edge (value 'to'); and a column named edge_id, containing a unique identifier for each edge. The edge_id is required for the plotting routine in order to identify which edge source should be connected with which edge destination.

Author(s)

Pepijn de Vries

Examples

```
data("ecosystem_services")

ecosystem_services_p1 <-
  pivot_stages_longer(
    data        = ecosystem_services,
    stages_from = c("activity_type", "pressure_cat",
                    "biotic_group", "service_division"),
    values_from = "RCSES")
```

```
## suppose we want to decorate our Sankey
## with information on the 'section' of the services:
ecosystem_services_p2 <-
  pivot_stages_longer(
    data        = ecosystem_services,
    stages_from = c("activity_type", "pressure_cat",
                    "biotic_group", "service_division"),
    values_from = "RCSES",
    additional_aes_from = "service_section")
```

PositionSankey *Position nodes and edges in a Sankey diagram*

Description

Calculates the x and y positions of elements (nodes and edges) in a Sankey diagram.

Usage

```
PositionSankey

position_sankey(
  width = "auto",
  align = c("bottom", "top", "center", "justify"),
  order = c("ascending", "descending", "as_is"),
  h_space = "auto",
  v_space = 0,
  nudge_x = 0,
  nudge_y = 0,
  split_nodes = FALSE,
  split_tol = 0.001,
  direction = c("forward", "backward"),
  ...
)
```

Arguments

width	Width of the node (numeric). When split_nodes is set to TRUE each part of the split node will have half this width. Use "auto" to automatically determine a suitable width.
align	A character that indicates how the nodes across the stages are aligned. It can be any of "top", "bottom", "center" or "justify".
order	A character indicating the method to be used for the order of stacking nodes and edges in a plot. Should be one of: ascending (default), sorts nodes and edges from large to small (largest on top); descending sorts nodes and edges from small to large (smallest on top); as_is will leave the order of nodes and edges as they are in data.

h_space	Horizontal space between split nodes (numeric). This argument is ignored when split_nodes == FALSE. Use "auto" to automatically position split nodes.
v_space	Vertical space between nodes (numeric). When set to zero (0), the Sankey diagram becomes an alluvial plot. Use "auto" to automatically determine a suitable vertical space.
nudge_x, nudge_y	Horizontal and vertical adjustment to nudge items by. Can be useful for offsetting labels.
split_nodes	A logical value indicating whether the source and destination nodes should be depicted as separate boxes.
split_tol	When the relative node size (resulting source and destination edges) differs more than this fraction, the node will be displayed as two separate bars.
direction	One of "forward" (default) or "backward". When set to "backward" the direction of the edges will be inverted. In most cases this parameter won't affect the plot. It can be helpful when you want to decorate the end of an edge (instead of the start; see examples).
...	Arguments passed on to ggplot2::ggproto().

Format

An object of class PositionSankey (inherits from Position, ggproto, gg) of length 13.

Details

Based on the stat_* function applied to the parent's (stat_sankeynode(), stat_sankeyedge) object either node or edge positions are calculated respectively. These positions can be used to add additional layers (e.g., text or labels) to the plot.

Value

Returns a ggplot2::Position class object.

Author(s)

Pepijn de Vries

Examples

```
library(ggplot2)
data("ecosystem_services")

pos  <- position_sankey(v_space = "auto", order = "ascending")
pos2 <- position_sankey(v_space = "auto", order = "ascending", direction = "backward")

## Let's subset the data, to make the plot less cluttered:
es_subset <- pivot_stages_longer(
  subset(ecosystem_services, RCSES > 0.01),
  c("activity_realm", "biotic_realm", "service_section"),
  "RCSES",
```

```
  "service_section"
)


plot <-
  ggplot(es_subset, aes(x = stage, y = RCSES, group = node,
                                  connector = connector, edge_id = edge_id,
                                  fill = node)) +
  geom_sankeynode(position = pos) +
  geom_sankeyedge(position = pos, aes(fill = service_section))

# position labels at nodes
plot + geom_text(aes(label = node), stat = "sankeynode", position = pos)
# position labels at the start of edges
plot + geom_text(aes(label = sprintf("%0.2f", RCSES)), stat = "sankeyedge", position = pos)
# position labels at the end of edges
plot + geom_text(aes(label = sprintf("%0.2f", RCSES)), stat = "sankeyedge", position = pos2)
```

`scale_waist_continuous` *Sankey edge waist line scales*

Description

The waist scale can be used to control the waist (i.e., the width of the edge at its centre) of edges in Sankey diagrams, in order to put emphasis on specific edges.

Usage

```
scale_waist_continuous(..., range = c(0, 1))

scale_waist_datetime(..., range = c(0, 1))

scale_waist_binned(..., range = c(0, 1))

scale_waist_discrete(..., range = c(0, 1))

scale_waist_manual(..., values = NULL, breaks = ggplot2::waiver())

scale_waist_identity(..., guide = "none")
```

Arguments

`...`	arguments passed onto underpinning scale constructors.
`range`	A `vector` of two `numeric` values used to scale the waist in between. Should be `>= 0`.

values — a set of aesthetic values to map data values to. The values will be matched in order (usually alphabetical) with the limits of the scale, or with breaks if provided. If this is a named vector, then the values will be matched based on the names instead. Data values that don't match will be given na.value.

breaks — One of:

- NULL for no breaks
- waiver() for the default breaks (the scale limits)
- A character vector of breaks
- A function that takes the limits as input and returns breaks as output

guide — Guide to use for this scale. Defaults to "none".

Details

This scale can be used to scale the centre of a Sankey edge. At one end of the scale the edge will be shaped like an hour glass, at the other end it will be shaped as an apple.

Value

Returns a ggplot2::Scale object which can be added to a ggplot2::ggplot to control the waist of Sankey diagram edges.

Author(s)

Pepijn de Vries

Examples

```
if (requireNamespace("ggplot2")) {
  library(ggplot2)
  data("ecosystem_services")

  p <-
    ggplot(ecosystem_services_pivot1, aes(x = stage, y = RCSES, group = node,
                                          connector = connector,
                                          edge_id = edge_id,
                                          waist = RCSES)) +
   geom_sankeyedge(v_space = "auto", ncp = 10) +
    geom_sankeynode(v_space = "auto")

  p + scale_waist_binned(range = c(0.1, 2))
  p + scale_waist_binned(range = c(2, 0.1))
}
```

StatSankeyedge *Sankey stats*

Description

Aggregates value on the y axis per group for nodes, and for all used aesthetics for edges.

Usage

```
StatSankeyedge

stat_sankeyedge(
  mapping = NULL,
  data = NULL,
  geom = "sankeyedge",
  position = "sankey",
  na.rm = FALSE,
  slope = 0.5,
  ncp = 100,
  show.legend = NA,
  inherit.aes = TRUE,
  ...
)

StatSankeynode

stat_sankeynode(
  mapping = NULL,
  data = NULL,
  geom = "sankeynode",
  position = "sankey",
  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	a string naming the ggplot2::proto Geom subclass. Should be either "sankeynode" or "sankeedge".
position	A character string or function specifying the positioning routine. By default this is "sankey".
na.rm	If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.
slope	Slope parameter (numeric) for the Bezier curves used to depict the edges. Any value between 0 and 1 will work nicely. Other non-zero values will also work.
ncp	Number of control points on the Bezier curve that forms the edge. Larger numbers will result in smoother curves, but cost more computational time. Default is 100.
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().
...	Passed to ggplot2::layer() function

Format

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

An object of class StatSankeynode (inherits from Stat, ggproto, gg) of length 3.

Details

Wrangles data before it can be passed to position_sankey().

Value

Returns a ggplot2 stat layer which can be used in a ggplot.

Author(s)

Pepijn de Vries

Examples

```
library(ggplot2)
data("ecosystem_services")

p <- ggplot(ecosystem_services_pivot1, aes(x = stage, y = RCSES, group = node,
                                           connector = connector, edge_id = edge_id,
                                           fill = node))
p + stat_sankeynode()
p + stat_sankeyedge()
```

Index