Package 'ddplot'

November 4, 2026

Type Package

Title Create D3 Based SVG Graphics

Version 0.2.0

Maintainer Mohamed El Fodil Ihaddaden <ihaddaden.fodeil@gmail.com>

Description Create 'D3' based 'SVG' ('Scalable Vector Graphics') graphics using a simple 'R' API.
The package aims to simplify
the creation of many 'SVG' plot types using a straightforward 'R' API.
The package relies on the 'r2d3' 'R' package and the 'D3' 'JavaScript' library.
See <https://rstudio.github.io/r2d3/> and <https://d3js.org/> respectively.

License GPL (>= 3)

Encoding UTF-8

URL https://feddelegrand7.github.io/ddplot/,
https://github.com/feddelegrand7/ddplot

BugReports https://github.com/feddelegrand7/ddplot/issues

RoxygenNote 7.3.2

Imports r2d3

Suggests knitr, rmarkdown, ggplot2, dplyr, tidyr, zoo, gapminder

VignetteBuilder knitr

NeedsCompilation no

Author Mohamed El Fodil Ihaddaden [aut, cre],
June Choe [ctb, cph],
Mike Bostock [ctb, cph] (D3.js developer),
RStudio [ctb, cph] (developers of the r2d3 package)

Repository CRAN

Date/Publication 2026-11-04 22:00:11 UTC

Contents

animated_histogram . . . 3
anim_line_chart . . . 4
area_band . . . 6
area_chart . . . 8
bar_chart . . . 10
bar_chart_race . . . 12
beeswarm_plot . . . 14
bullet_chart . . . 17
bump_chart . . . 19
calendar_heatmap . . . 21
chord_diagram . . . 25
dumbbell_chart . . . 28
flame . . . 31
flower . . . 32
gauge_chart . . . 33
glass_fill . . . 35
heart_fill . . . 36
hexbin_plot . . . 38
histogram . . . 41
horz_bar_chart . . . 43
horz_lollipop . . . 45
line_chart . . . 46
liquid_chart . . . 52
lollipop_chart . . . 53
mosaic_chart . . . 55
parliament_chart . . . 59
pie_chart . . . 60
plant_growth . . . 62
pulse_grid . . . 63
radar_chart . . . 65
radial_bar_chart . . . 68
ridgeline_plot . . . 71
rose_chart . . . 74
scatter_plot . . . 77
slope_chart . . . 79
stacked_area_chart . . . 82
streamgraph . . . 84
sunburst_chart . . . 88
treemap_chart . . . 91
violin_plot . . . 94
waffle_chart . . . 98
waterfall_chart . . . 100

Index **104**

`animated_histogram` *Create an animated histogram.*

Description

Create an animated histogram.

Usage

```
animated_histogram(
  x,
  bins = 30,
  duration = 2000,
  delay = 100,
  fill = "crimson",
  xFontSize = 10,
  yFontSize = 10,
  xticks = NULL,
  yticks = NULL,
  xtitle = NULL,
  xtitleFontSize = 16,
  ytitle = NULL,
  ytitleFontSize = 16,
  title = NULL,
  titleFontSize = 22,
  stroke = "crimson",
  strokeWidth = NULL,
  font = "Verdana, Geneva, Tahoma, sans-serif",
  bgcol = "#CAD0D3",
  opacity = 1,
  axisCol = "black",
  width = NULL,
  height = NULL
)
```

Arguments

`x`	A vector of data.
`bins`	The number of bins to consider. Defaults to 30.
`duration`	The duration of the bars' transition in milliseconds. Defaults to 2000.
`delay`	The amount of time (in milliseconds) that precedes before triggering the appearance of each bar. Defaults to 100.
`fill`	The color of the bars. Defaults to 'crimson'.
`xFontSize`	the font size of the x-axis labels. Defaults to 10.
`yFontSize`	the font size of the y-axis labels. Defaults to 10.

xticks	Optional. the number of x-axis ticks to consider.
yticks	Optional. The number of y-axis ticks to consider.
xtitle	Optional. The title of the x-axis.
xtitleFontSize	The font size of the x-axis title. Defaults to 16.
ytitle	Optional. The title of the y-axis.
ytitleFontSize	The font size of the y-axis title. Defaults to 16.
title	Optional. The title of the plot.
titleFontSize	The font size of the plot title. Defaults to 22.
stroke	The stroke color of the bars. Defaults to 'crimson'.
strokeWidth	Optional. the stroke width of the bars.
font	The font family to consider for the titles. Defaults to "Verdana, Geneva, Tahoma, sans-serif".
bgcol	The background color of the SVG. Defaults to "#CAD0D3" HEX color.
opacity	The color opacity of the bars (from 0 to 1). Defaults to 1.
axisCol	the color of the x and y axis. It includes the ticks, the labels and titles. Defaults to 'black'.
width	Optional. The width of the SVG output.
height	Optional. The height of the SVG output.

Value

An animated SVG histogram.

Examples

```
animated_histogram(
 x = mtcars$mpg,
 duration = 2000,
 delay = 100
)
```

---

anim_line_chart *Create an animated line chart*

Description

Create an animated line chart

Usage

```
anim_line_chart(
  data,
  x,
  y,
  curve = "curveLinear",
  duration = 5000,
  stroke = "crimson",
  strokeWidth = 1.5,
  xticks = NULL,
  yticks = NULL,
  xtitle = NULL,
  xtitleFontSize = 16,
  ytitle = NULL,
  ytitleFontSize = 16,
  title = NULL,
  titleFontSize = 22,
  font = "Verdana, Geneva, Tahoma, sans-serif",
  bgcol = "#CAD0D3",
  opacity = 1,
  axisCol = "black",
  width = NULL,
  height = NULL
)
```

Arguments

data	The data frame containing the variables to consider.
x	The x-variable to consider. Must be a date variable in 'yyyy-mm-dd' format.
y	The y-variable to consider.
curve	Optional. The line's curve type to render. A complete list can be found here <https://github.com/d3/d3-shape#curves>. Defaults to 'curveLinear'.
duration	The duration in Milliseconds of the animation. Defaults to 5000.
stroke	The color of the line. Defaults to 'crimson'.
strokeWidth	The width of the line. Defaults to 1.5.
xticks	Optional. the number of x-axis ticks to consider.
yticks	Optional. The number of y-axis ticks to consider.
xtitle	Optional. The title of the x-axis.
xtitleFontSize	The font size of the x-axis title. Defaults to 16.
ytitle	Optional. The title of the y-axis.
ytitleFontSize	The font size of the y-axis title. Defaults to 16.
title	Optional. The title of the plot.
titleFontSize	The font size of the plot title. Defaults to 22.

font	The font family to consider for the titles. Defaults to "Verdana, Geneva, Tahoma, sans-serif".
bgcol	The background color of the SVG. Defaults to "#CAD0D3" HEX color.
opacity	The color opacity of the bars (from 0 to 1). Defaults to 1.
axisCol	the color of the x and y axis. It includes the ticks, the labels and titles. Defaults to 'black'.
width	Optional. The width of the SVG output.
height	Optional. The height of the SVG output.

Value

An animated SVG line chart.

Examples

```
airpassengers <- data.frame(
 passengers = as.matrix(AirPassengers),
 date= zoo::as.Date(time(AirPassengers))
)
anim_line_chart(
 data = airpassengers,
 x = "date",
 y = "passengers",
 duration = 10000 # in milliseconds (10 seconds)
)
```

---

area_band *Create a band chart*

Description

Create a band chart

Usage

```
area_band(
  data,
  x,
  yLower,
  yUpper,
  fill = "crimson",
  stroke = NULL,
  strokeWidth = NULL,
  xticks = NULL,
  yticks = NULL,
  xtitle = NULL,
  xtitleFontSize = 16,
```

```
  ytitle = NULL,
  ytitleFontSize = 16,
  title = NULL,
  titleFontSize = 22,
  font = "Verdana, Geneva, Tahoma, sans-serif",
  bgcol = "#CAD0D3",
  opacity = 1,
  axisCol = "black",
  width = NULL,
  height = NULL
)
```

Arguments

data	The data frame containing the variables to consider.
x	The x-variable to consider. Must be a date variable in 'yyyy-mm-dd' format.
yLower	The y-lower band variable to consider.
yUpper	The y-upper band variable to consider.
fill	The color of the band. Defaults to 'crimson'.
stroke	Optional. The color of the stroke of the band.
strokeWidth	Optional. The width of the band stroke.
xticks	Optional. the number of x-axis ticks to consider.
yticks	Optional. The number of y-axis ticks to consider.
xtitle	Optional. The title of the x-axis.
xtitleFontSize	The font size of the x-axis title. Defaults to 16.
ytitle	Optional. The title of the y-axis.
ytitleFontSize	The font size of the y-axis title. Defaults to 16.
title	Optional. The title of the plot.
titleFontSize	The font size of the plot title. Defaults to 22.
font	The font family to consider for the titles. Defaults to "Verdana, Geneva, Tahoma, sans-serif".
bgcol	The background color of the SVG. Defaults to "#CAD0D3" HEX color.
opacity	The color opacity of the area chart (from 0 to 1). Defaults to 1.
axisCol	the color of the x and y axis. It includes the ticks, the labels and titles. Defaults to 'black'.
width	Optional. The width of the SVG output.
height	Optional. The height of the SVG output.

Value

A SVG band chart

Examples

```
airpassengers <- data.frame(
 passengers_lower = as.matrix(AirPassengers),
 passengers_upper = as.matrix(AirPassengers) + 40,
 date= zoo::as.Date(time(AirPassengers))
)

area_band(
 data = airpassengers,
 x = "date",
 yLower = "passengers_lower",
 yUpper = "passengers_upper",
 fill = "yellow",
 stroke = "black"
)
```

---

area_chart *Create an area chart*

Description

Create an area chart

Usage

```
area_chart(
  data,
  x,
  y,
  fill = "crimson",
  stroke = NULL,
  strokeWidth = NULL,
  xticks = NULL,
  yticks = NULL,
  xtitle = NULL,
  xtitleFontSize = 16,
  ytitle = NULL,
  ytitleFontSize = 16,
  title = NULL,
  titleFontSize = 22,
  font = "Verdana, Geneva, Tahoma, sans-serif",
  bgcol = "#CAD0D3",
  opacity = 1,
  axisCol = "black",
  width = NULL,
  height = NULL
)
```

Arguments

data	The data frame containing the variables to consider.
x	The x-variable to consider. Must be a date variable in 'yyyy-mm-dd' format.
y	The y-variable to consider.
fill	The color of the area chart. Defaults to 'crimson'.
stroke	Optional. The color of the stroke of the area.
strokeWidth	Optional. The width of the area stroke.
xticks	Optional. the number of x-axis ticks to consider.
yticks	Optional. The number of y-axis ticks to consider.
xtitle	Optional. The title of the x-axis.
xtitleFontSize	The font size of the x-axis title. Defaults to 16.
ytitle	Optional. The title of the y-axis.
ytitleFontSize	The font size of the y-axis title. Defaults to 16.
title	Optional. The title of the plot.
titleFontSize	The font size of the plot title. Defaults to 22.
font	The font family to consider for the titles. Defaults to "Verdana, Geneva, Tahoma, sans-serif".
bgcol	The background color of the SVG. Defaults to "#CAD0D3" HEX color.
opacity	The color opacity of the area chart (from 0 to 1). Defaults to 1.
axisCol	the color of the x and y axis. It includes the ticks, the labels and titles. Defaults to 'black'.
width	Optional. The width of the SVG output.
height	Optional. The height of the SVG output.

Value

a SVG area chart

Examples

```
# 1. converting AirPassengers to a tidy data frame
airpassengers <- data.frame(
  passengers = as.matrix(AirPassengers),
  date= zoo::as.Date(time(AirPassengers))
)

# 2. plotting the area chart
area_chart(
  data = airpassengers,
  x = "date",
  y = "passengers",
  fill = "purple",
  bgcol = "white"
)
```

bar_chart *Create a bar chart.*

Description

Create a bar chart.

Usage

```
bar_chart(
  data,
  x,
  y,
  fill = "crimson",
  sort = "none",
  paddingWidth = 0.1,
  xticks = NULL,
  xFontSize = 10,
  yFontSize = 10,
  yticks = NULL,
  xtitle = NULL,
  xtitleFontSize = 16,
  ytitle = NULL,
  ytitleFontSize = 16,
  title = NULL,
  titleFontSize = 22,
  stroke = "crimson",
  strokeWidth = NULL,
  font = "Verdana, Geneva, Tahoma, sans-serif",
  bgcol = "#CAD0D3",
  opacity = 1,
  axisCol = "black",
  width = NULL,
  height = NULL
)
```

Arguments

data	The data frame containing the variables to consider.
x	The x-variable to consider.
y	The y-variable to consider.
fill	The color of the bars. Defaults to 'crimson'.
sort	Whether to sort or not the bars. Takes three values 'none' which is the default, 'ascending' or 'descending'.
paddingWidth	The distance between each bar. The value goes from 0 to 0.99 included. Defaults to 0.1.

xticks	Optional. the number of x-axis ticks to consider.
xFontSize	the font size of the x-axis labels. Defaults to 10.
yFontSize	the font size of the y-axis labels. Defaults to 10.
yticks	Optional. The number of y-axis ticks to consider.
xtitle	Optional. The title of the x-axis.
xtitleFontSize	The font size of the x-axis title. Defaults to 16.
ytitle	Optional. The title of the y-axis.
ytitleFontSize	The font size of the y-axis title. Defaults to 16.
title	Optional. The title of the plot.
titleFontSize	The font size of the plot title. Defaults to 22.
stroke	The stroke color of the bars. Defaults to 'crimson'.
strokeWidth	Optional. the stroke width of the bars.
font	The font family to consider for the titles. Defaults to "Verdana, Geneva, Tahoma, sans-serif".
bgcol	The background color of the SVG. Defaults to "#CAD0D3" HEX color.
opacity	The color opacity of the bars (from 0 to 1). Defaults to 1.
axisCol	the color of the x and y axis. It includes the ticks, the labels and titles. Defaults to 'black'.
width	Optional. The width of the SVG output.
height	Optional. The height of the SVG output.

Value

A SVG bar chart.

Examples

```
library(ggplot2) #needed for the mpg data frame
library(dplyr) #needed for data wrangling

mpg %>% group_by(manufacturer) %>%
 summarise(mean_cty = mean(cty)) %>%
 bar_chart(
   x = "manufacturer",
   y = "mean_cty",
   sort = "ascending",
   xFontSize = 10,
   yFontSize = 10,
   fill = "orange",
   strokeWidth = 1,
   ytitle = "average cty value",
   title = "Average City Miles per Gallon by manufacturer",
   titleFontSize = 16
 )
```

bar_chart_race *Create a bar chart race.*

Description

Create a bar chart race.

Usage

```
bar_chart_race(
  data,
  x,
  y,
  time,
  ease = "Linear",
  frameDur = 500,
  transitionDur = 500,
  colorCategory = "Accent",
  sort = "descending",
  paddingWidth = 0.1,
  xFontSize = 10,
  yFontSize = 10,
  xticks = 10,
  xtitle = NULL,
  xtitleFontSize = 16,
  ytitle = NULL,
  ytitleFontSize = 14,
  title = NULL,
  titleFontSize = 22,
  stroke = "black",
  strokeWidth = NULL,
  font = "Verdana, Geneva, Tahoma, sans-serif",
  bgcol = "#CAD0D3",
  panelcol = "#EBEBEBFF",
  xgridlinecol = "white",
  opacity = 1,
  timeLabel = TRUE,
 timeLabelOpts = list(size = 32, prefix = "", suffix = "", xOffset = 0.5, yOffset = 1),
  width = NULL,
  height = NULL
)
```

Arguments

data	The data frame containing the variables to consider.
x	The x-variable to consider.

y	The y-variable to consider.
time	The time variable to consider.
ease	The easing method, you can find more here <https://github.com/d3/d3-ease>. Defaults to 'Linear'.
frameDur	The time spent paused on each frame (time point) in milliseconds.
transitionDur	The time spent transitioning between frames in milliseconds.
colorCategory	A D3 categorical color scheme, you can find more here <https://github.com/d3/d3-scale-chromatic#categorical>. Defaults to 'Accent'.
sort	Whether to sort or not the bars. Takes three values 'none' which is the default, 'ascending' or 'descending'. Defaults to 'descending'.
paddingWidth	The distance between each bar. The value goes from 0 to 0.99 included. Defaults to 0.1.
xFontSize	the font size of the x-axis labels. Defaults to 10.
yFontSize	the font size of the y-axis labels. Defaults to 10.
xticks	the number of y-axis ticks to consider. Defaults to 10.
xtitle	Optional. The title of the x-axis.
xtitleFontSize	The font size of the x-axis title. Defaults to 16.
ytitle	Optional. The title of the y-axis.
ytitleFontSize	The font size of the y-axis title. Defaults to 14.
title	Optional. The title of the plot.
titleFontSize	The font size of the plot title. Defaults to 22.
stroke	The stroke color of the bars. Defaults to 'black'.
strokeWidth	Optional. the stroke width of the bars.
font	The font family to consider for the titles. Defaults to "Verdana, Geneva, Tahoma, sans-serif".
bgcol	The background color of the SVG. Defaults to "#CAD0D3" HEX color.
panelcol	The background color of the panel. Defaults to "#EBEBEBFF" HEX color.
xgridlinecol	The color of the x-axis grid lines. Defaults to 'white'.
opacity	The color opacity of the bars (from 0 to 1). Defaults to 1.
timeLabel	Whether to show a label for the value of the time variable. Defaults to TRUE.
timeLabelOpts	Options for labeling the value of the time variable. Takes a list specifying the 'size', 'prefix', 'suffix', 'xOffset', and 'yOffset'. Offsets are scaled relative to the dimensions of the label, from the bottom-right corner of the panel.
width	Optional. The width of the SVG output.
height	Optional. The height of the SVG output.

Value

An animated SVG bar chart race, wrapped in a div.

Examples

```
library(gapminder)
library(dplyr)
# let's select a set of countries only
gapminder <- gapminder %>%
filter(
country %in% c("Algeria", "Mexico", "Iceland", "Greece", "Finland")
)

bar_chart_race(
data = gapminder,
x = "lifeExp",
y = "country",
time = "year",
ytitle = "Country",
xtitle = "Life expectancy",
title = "Bar chart race of countries life expectancy"
)
```

`beeswarm_plot` *Create a beeswarm plot.*

Description

A beeswarm plot displays individual data points along a numeric axis, spreading them out so they do not overlap (like a swarm of bees). It is a transparent alternative to box plots or violin plots because every observation is visible. When a grouping variable is supplied, one swarm is drawn per group along the opposite axis.

The layout is computed with a D3 force simulation that pulls each point toward its true value on the x-axis while resolving collisions so no two circles overlap.

The plot is interactive: hovering a point highlights it while dimming the rest of the swarm, and (when `tooltip` columns are supplied) shows a tooltip with those columns' values. When a `group` is supplied, hovering a group label focuses that swarm and clicking a label toggles it on or off. An optional mean/median reference line can be drawn per swarm. The interactive behaviors are controlled by `tooltip`, `hover`, `animate`, and `stat`.

Usage

```
beeswarm_plot(
  data,
  x,
  group = NULL,
  tooltip = NULL,
  hover = TRUE,
  animate = TRUE,
  stat = c("none", "mean", "median"),
  statColor = "#d62728",
```

```
  col = "steelblue",
  colorPalette = "Tableau10",
  radius = 4,
  opacity = 0.75,
  stroke = "white",
  strokeWidth = 0.5,
  xtitle = NULL,
  xtitleFontSize = 13,
  title = NULL,
  titleFontSize = 16,
  xFontSize = 11,
  yFontSize = 11,
  font = "Verdana, Geneva, Tahoma, sans-serif",
  bgcol = "white",
  axisCol = "black",
  width = NULL,
  height = NULL
)
```

Arguments

`data`	The data frame containing the variables to consider.
`x`	The name of the numeric column whose distribution is displayed.
`group`	Optional. The name of a categorical column used to split the data into separate swarms. Defaults to `NULL` (single swarm).
`tooltip`	Optional. A column name, or a vector of column names, whose values are shown in a tooltip when a point is hovered. Each column appears on its own line as `"column: value"`. When `NULL` (default) no tooltip is shown.
`hover`	Logical. When `TRUE` (default), hovering a point enlarges it and dims the rest of the swarm to make it stand out.
`animate`	Logical. When `TRUE` (default), points grow into place with a short staggered entry animation on first render.
`stat`	Character. Draws a reference line per swarm at the `"mean"` or `"median"` of the values. Defaults to `"none"` (no line).
`statColor`	Color of the mean/median reference line and its label. Defaults to `"#d62728"`.
`col`	Fill color of the points when no `group` is supplied. Defaults to `"steelblue"`.
`colorPalette`	D3 categorical color scheme used when `group` is supplied (e.g. `"Tableau10"`, `"Category10"`, `"Set2"`). Defaults to `"Tableau10"`.
`radius`	Radius of each point in pixels. Defaults to `4`.
`opacity`	Fill opacity of the points (0 to 1). Defaults to `0.75`.
`stroke`	Stroke color of the points. Defaults to `"white"`.
`strokeWidth`	Stroke width in pixels. Defaults to `0.5`.
`xtitle`	Optional. Title of the x-axis.
`xtitleFontSize`	Font size of the x-axis title. Defaults to `13`.

title	Optional. Title of the chart.
titleFontSize	Font size of the chart title. Defaults to 16.
xFontSize	Font size of the x-axis tick labels. Defaults to 11.
yFontSize	Font size of the y-axis group labels. Defaults to 11.
font	The font family used for all text. Defaults to "Verdana, Geneva, Tahoma, sans-serif".
bgcol	Background color of the SVG. Defaults to "white".
axisCol	Color of axis lines, ticks, and labels. Defaults to "black".
width	Optional. The width of the SVG output.
height	Optional. The height of the SVG output.

Value

An SVG beeswarm plot.

Examples

```
library(ggplot2) # for the mpg dataset

# Single swarm — distribution of highway fuel economy
beeswarm_plot(
  data = mpg,
  x    = "hwy",
  col  = "steelblue",
  xtitle = "Highway miles per gallon",
  title  = "Distribution of hwy"
)

# Grouped swarm — hwy by vehicle class
beeswarm_plot(
  data  = mpg,
  x     = "hwy",
  group = "class",
  xtitle = "Highway miles per gallon",
  title  = "Highway fuel economy by vehicle class"
)

# Grouped swarm on the iris dataset
beeswarm_plot(
  data         = iris,
  x            = "Sepal.Length",
  group        = "Species",
  colorPalette = "Set2",
  radius       = 5,
  xtitle       = "Sepal length (cm)",
  title        = "Sepal length by species"
)

# Interactive extras: median line per swarm and a tooltip label column
```

```
beeswarm_plot(
  data    = mpg,
  x       = "hwy",
  group   = "class",
  tooltip = "model",
  stat    = "median",
  xtitle  = "Highway miles per gallon",
  title   = "Highway fuel economy by vehicle class"
)
```

`bullet_chart` *Create a bullet chart.*

Description

A bullet chart displays a single quantitative measure (the actual value) against qualitative range bands (e.g. poor / satisfactory / good) and an optional target marker. It was designed by Stephen Few as a compact, information-dense alternative to gauge charts.

Usage

```
bullet_chart(
  value,
  ranges,
  target = NULL,
  rangeColors = NULL,
  title = "",
  subtitle = "",
  min = 0,
  valueColor = "#1a1a2e",
  targetColor = "#e63946",
  tickCount = 5,
  fontSize = 12,
  font = "Verdana, Geneva, Tahoma, sans-serif",
  bgcol = "white",
  width = NULL,
  height = NULL
)
```

Arguments

`value`	Numeric. The actual value to display (the main bar).
`ranges`	Numeric vector of upper bounds for each qualitative band, in ascending order (e.g. `c(40, 70, 100)`). The lower bound of the first band is `min`.
`target`	Optional numeric. The target / goal value shown as a short vertical marker. Defaults to `NULL` (no marker drawn).

rangeColors	Character vector of colors for each band, in the same order as ranges. Defaults to a sequence of grays from darker (low range) to lighter (high range).
title	Character. Label displayed to the left of the chart. Defaults to "".
subtitle	Character. Smaller label displayed below title. Defaults to "".
min	Numeric. The minimum value of the scale. Defaults to 0.
valueColor	The color of the actual-value bar. Defaults to "#1a1a2e".
targetColor	The color of the target marker. Defaults to "#e63946".
tickCount	Number of axis ticks. Defaults to 5.
fontSize	Base font size in pixels. Defaults to 12.
font	The font family used for all text. Defaults to "Verdana, Geneva, Tahoma, sans-serif".
bgcol	The background color of the SVG. Defaults to "white".
width	Optional. The width of the SVG output.
height	Optional. The height of the SVG output.

Value

An SVG bullet chart.

Examples

```
# Sales revenue vs target
bullet_chart(
  value  = 270,
  target = 300,
  ranges = c(150, 225, 350),
  title  = "Revenue",
  subtitle = "USD thousands"
)

# Customer satisfaction score
bullet_chart(
  value  = 7.4,
  target = 8.0,
  ranges = c(4, 7, 10),
  title  = "Satisfaction",
  subtitle = "out of 10",
  valueColor = "steelblue"
)

# Server response time (lower is better — ranges go poor → good left to right)
bullet_chart(
  value       = 320,
  target      = 250,
  ranges      = c(200, 500, 1000),
  rangeColors = c("#f5f5f5", "#e0e0e0", "#c0c0c0"),
  title       = "Response",
  subtitle    = "ms",
```

```
  min         = 0
)
```

---

bump_chart *Create a bump chart.*

Description

A bump chart shows how the *rank* of several entities changes across successive time periods. Each entity is drawn as a smooth line that moves up or down as it overtakes, or is overtaken by, the others, with a circle at every period showing its rank. Popularised by the Financial Times, The Economist and FiveThirtyEight (league tables, election polls, country rankings), it is the clearest way to tell a "who is ahead?" story when the order matters more than the exact values.

The data must be in long format: one row per entity and period. By default the ranks are computed per period from the numeric `y` column (rank 1 = highest value). Set `ranked = TRUE` if `y` already contains the ranks.

Usage

```
bump_chart(
  data,
  x,
  y,
  group,
  ranked = FALSE,
  rankOrder = "desc",
  colors = NULL,
  highlight = NULL,
  greyCol = "#d3d3d3",
  lineWidth = 3,
  circleRadius = 11,
  showRank = TRUE,
  title = NULL,
  titleFontSize = 16,
  xFontSize = 11,
  labelFontSize = 11,
  font = "Verdana, Geneva, Tahoma, sans-serif",
  bgcol = "white",
  axisCol = "black",
  width = NULL,
  height = NULL
)
```

Arguments

`data`	The data frame containing the variables to consider, in long format (one row per group and period).

x	The name of the column holding the time period. Numeric and Date columns are ordered chronologically, factors follow their level order and character columns follow their order of appearance.
y	The name of the numeric column holding the value to rank (or the rank itself when ranked = TRUE).
group	The name of the column identifying each entity (one line per entity). Each group can appear at most once per period.
ranked	Whether y already contains ranks. Defaults to FALSE, in which case ranks are computed per period from y.
rankOrder	How ranks are computed when ranked = FALSE. One of "desc" (default, the highest value gets rank 1) or "asc" (the lowest value gets rank 1). Ties are broken by order of appearance in the data.
colors	Optional. A character vector of colors, recycled across the groups in order of appearance, or a named vector mapping group names to colors. Colors must be valid CSS colors (e.g. "steelblue" or "#1f77b4"); R-only names such as "grey40" are not understood by the browser. Defaults to the D3 Tableau 10 palette.
highlight	Optional. A character vector of group names to highlight. All other groups are drawn in greyCol.
greyCol	Color used for groups that are not highlighted. Defaults to "#d3d3d3".
lineWidth	Width of the lines in pixels. Defaults to 3.
circleRadius	Maximum radius of the circles in pixels. It is reduced automatically when there are many groups or periods. Defaults to 11.
showRank	Whether to print the rank number inside each circle. Defaults to TRUE.
title	Optional. Title of the chart.
titleFontSize	Font size of the chart title. Defaults to 16.
xFontSize	Font size of the period labels on the x-axis. Defaults to 11.
labelFontSize	Font size of the group labels drawn at both ends of each line. Defaults to 11.
font	The font family used for all text. Defaults to "Verdana, Geneva, Tahoma, sans-serif".
bgcol	Background color of the SVG. Defaults to "white".
axisCol	Color of the period labels, grid lines and title. Defaults to "black".
width	Optional. The width of the SVG output.
height	Optional. The height of the SVG output.

Value

An SVG bump chart.

Examples

```
# Points scored by five teams over four seasons
league <- data.frame(
  season = rep(c("2023", "2024", "2025", "2026"), each = 5),
  team   = rep(c("Lions", "Tigers", "Bears", "Wolves", "Eagles"), 4),
  points = c(80, 75, 60, 55, 50,
             70, 82, 58, 66, 49,
             65, 78, 72, 69, 52,
             60, 74, 81, 70, 77)
)

bump_chart(
  data  = league,
  x     = "season",
  y     = "points",
  group = "team",
  title = "League ranking by season"
)

# GDP per capita rank of the Asian "tiger" economies, highlighting two
library(gapminder)

countries <- c("Japan", "Singapore", "Hong Kong, China", "Taiwan",
               "Korea, Rep.", "Malaysia", "Thailand", "China")
gdp <- gapminder[gapminder$country %in% countries &
                 gapminder$year >= 1967, ]
gdp$country <- as.character(gdp$country)

bump_chart(
  data      = gdp,
  x         = "year",
  y         = "gdpPercap",
  group     = "country",
  highlight = c("Singapore", "Korea, Rep."),
  title     = "GDP per capita rank, 1967-2007"
)
```

`calendar_heatmap` *Create a calendar heatmap.*

Description

A calendar heatmap lays out one rounded square per day on a grid where each column is a week and each row is a day of the week, so a whole year fits in a single compact band. Each square is colored by the value recorded on that day using a sequential palette, which makes weekly rhythms, seasonal patterns, streaks and gaps easy to spot. The layout was popularised by GitHub's contribution graph and is the signature chart of the Frappe Charts library.

Month labels are drawn above the grid, weekday labels on its left, and a "Less ... More" legend underneath. When the data span several years, one panel is stacked per year. Days without data

(and, when all values are non-negative, days with a value of zero) are drawn in `emptyCol`. Hovering over a day shows a tooltip with its value and date, and hovering over a legend square highlights the days that fall in that color bucket.

Usage

```
calendar_heatmap(
  data,
  date,
  value = NULL,
  aggregate = c("sum", "mean", "max", "min", "count"),
  weekStart = c("sunday", "monday"),
  colors = c("#c0ddf9", "#73b3f3", "#3886e1", "#17459e"),
  emptyCol = "#ebedf0",
  cellRadius = 2,
  cellGap = 3,
  valueLabel = NULL,
  valueFormat = ",.2~f",
  dateFormat = "%a %b %-d, %Y",
  showYearLabel = TRUE,
  showMonthLabels = TRUE,
  showWeekdayLabels = TRUE,
  labelFontSize = 10,
  legend = TRUE,
  legendLabels = c("Less", "More"),
  hover = TRUE,
  animate = TRUE,
  title = NULL,
  titleFontSize = 16,
  font = "Verdana, Geneva, Tahoma, sans-serif",
  bgcol = "white",
  axisCol = "#555555",
  width = NULL,
  height = NULL
)
```

Arguments

`data`	The data frame containing the variables to consider.
`date`	The name of the column holding the dates. It can be of class `Date`, `POSIXct` or a character vector in `"YYYY-MM-DD"` format.
`value`	Optional. The name of the numeric column holding the value of each day. When `NULL` (default), the number of rows per day is used, which is handy for event logs (one row per event).
`aggregate`	How to combine several rows that share the same date. One of `"sum"` (default), `"mean"`, `"max"`, `"min"` or `"count"`. Ignored (always `"count"`) when `value` is `NULL`.

weekStart	The first day of the week, i.e. the top row of the grid. Either "sunday" (default) or "monday".
colors	A vector of colors, from low to high, used to fill days that have a value. Values are split into length(colors) equal-width buckets. Defaults to a Frappe-like blue ramp c("#c0ddf9", "#73b3f3", "#3886e1", "#17459e").
emptyCol	Fill color of days without data (or with a value of zero when all values are non-negative). Defaults to "#ebedf0".
cellRadius	Corner radius of the day squares in pixels. Defaults to 2.
cellGap	Gap between day squares in pixels. Defaults to 3.
valueLabel	Optional. The unit shown after the value in the tooltip and year totals (e.g. "commits"). Defaults to the name of the value column, or "events" when value is NULL.
valueFormat	A d3-format specifier used to display values (e.g. ",.0f", "$,.2s"). Defaults to ",.2~f".
dateFormat	A d3-time-format specifier used to display dates in the tooltip. Defaults to "%a %b %-d, %Y" (e.g. "Wed Jan 7, 2026").
showYearLabel	Whether to print the year and its total above each panel. Defaults to TRUE.
showMonthLabels	Whether to print month labels above the grid. Defaults to TRUE.
showWeekdayLabels	Whether to print weekday labels (Mon, Wed, Fri) on the left of the grid. Defaults to TRUE.
labelFontSize	Font size of the month, weekday, year and legend labels. Defaults to 10.
legend	Whether to display the "Less ... More" legend. Defaults to TRUE.
legendLabels	A character vector of length 2 giving the text drawn on each side of the legend. Defaults to c("Less", "More").
hover	Whether to highlight the hovered day and show a tooltip. Defaults to TRUE.
animate	Whether the day squares fade in with a staggered animation on first render. Defaults to TRUE.
title	Optional. Title of the chart.
titleFontSize	Font size of the chart title. Defaults to 16.
font	The font family used for all text. Defaults to "Verdana, Geneva, Tahoma, sans-serif".
bgcol	Background color of the SVG. Defaults to "white".
axisCol	Color of the title and of the month, weekday, year and legend labels. Defaults to "#555555".
width	Optional. The width of the SVG output.
height	Optional. The height of the SVG output. Defaults to a height that fits the number of years displayed.

Value

An SVG calendar heatmap.

Examples

```
# One year of simulated daily activity, busier on weekdays
set.seed(2026)
days <- seq(as.Date("2026-01-01"), as.Date("2026-12-31"), by = "day")
weekday <- !(format(days, "%u") %in% c("6", "7"))
activity <- data.frame(
  day     = days,
  commits = rpois(length(days), lambda = ifelse(weekday, 6, 1.5)) *
            rbinom(length(days), 1, 0.8)
)

calendar_heatmap(
  data       = activity,
  date       = "day",
  value      = "commits",
  title      = "Commits in 2026"
)

# Several years stacked, weeks starting on Monday, GitHub-like greens
set.seed(1)
days <- seq(as.Date("2021-01-01"), as.Date("2023-12-31"), by = "day")
season <- 10 + 8 * sin(2 * pi * as.numeric(format(days, "%j")) / 365)
rainfall <- data.frame(day = days, rain = pmax(0, rnorm(length(days), season, 6)))

calendar_heatmap(
  data        = rainfall,
  date        = "day",
  value       = "rain",
  weekStart   = "monday",
  colors      = c("#9be9a8", "#40c463", "#30a14e", "#216e39"),
  valueLabel  = "mm of rain",
  valueFormat = ",.1f",
  title       = "Daily rainfall, 2021-2023"
)

# Event log: one row per event, counted per day
set.seed(7)
events <- data.frame(
  when = as.Date("2025-01-01") + sample(0:364, 900, replace = TRUE)
)

calendar_heatmap(
  data       = events,
  date       = "when",
  valueLabel = "orders",
  colors     = c("#fde0c5", "#facba6", "#f59e72", "#e8603c", "#c0392b"),
  emptyCol   = "#f2f2f2",
  cellRadius = 3,
  title      = "Orders per day"
)
```

chord_diagram *Create a chord diagram.*

Description

A chord diagram shows the flows (or connections) between a set of entities arranged around a circle. Each entity is drawn as an outer arc whose length is proportional to the total flow it is involved in, and each pair of entities is linked by a ribbon whose width at either end is proportional to the flow in that direction. Chord diagrams are a compact way to display migration, trade, transitions between states, or any other square matrix of flows.

The look and feel are inspired by the chord chart of the Nivo visualization library, reimplemented in plain D3. The chart is interactive: hovering an arc (or its label) keeps the ribbons connected to that entity and fades out all the others, while a tooltip lists the entity's totals and its largest flows. Hovering a ribbon isolates it and shows the flow in both directions. Clicking an arc pins its highlight; clicking it again, or clicking the background, releases it.

Usage

```
chord_diagram(
  data,
  from = "from",
  to = "to",
  value = "value",
  directed = TRUE,
  colors = NULL,
  colorPalette = "Tableau10",
  ribbonColor = c("source", "target", "gradient"),
  sortGroups = c("none", "descending", "ascending"),
  padAngle = 0.03,
  innerRadiusRatio = 0.92,
  arcOpacity = 1,
  ribbonOpacity = 0.6,
  fadeOpacity = 0.1,
  strokeWidth = 0.5,
  showLabels = TRUE,
  rotateLabels = FALSE,
  labelFontSize = 11,
  labelOffset = 10,
  valueFormat = ",~f",
  hover = TRUE,
  animate = TRUE,
  title = NULL,
  titleFontSize = 16,
  font = "Verdana, Geneva, Tahoma, sans-serif",
  bgcol = "white",
  textCol = "black",
```

```
  width = NULL,
  height = NULL
)
```

Arguments

data	Either a data frame in edge-list format (one row per flow, with the columns named by from, to and value), or a square numeric matrix of flows with row and/or column names, where data[i, j] is the flow from entity i to entity j. Duplicated edges are summed. Self-flows (from == to) are allowed.
from	The name of the column holding the origin of each flow. Only used when data is a data frame. Defaults to "from".
to	The name of the column holding the destination of each flow. Only used when data is a data frame. Defaults to "to".
value	The name of the numeric (non-negative) column holding the size of each flow. Use NULL to count rows (every row counts as 1). Only used when data is a data frame. Defaults to "value".
directed	Whether flows are directed. When TRUE (default), A -> B and B -> A are kept apart, so a ribbon can be wider at one end than at the other. When FALSE, flows are treated as undirected links: in an edge list, both directions are summed; a symmetric matrix is used as is, and a non-symmetric one is symmetrized by adding its transpose (the diagonal is left unchanged).
colors	Optional. A vector of colors for the entities, recycled if needed. A named vector maps colors to entity names (unnamed entities fall back to colorPalette). Defaults to NULL.
colorPalette	D3 categorical color scheme used when colors is NULL (e.g. "Tableau10", "Category10", "Set2", "Dark2", "Paired"). Defaults to "Tableau10".
ribbonColor	How ribbons are colored: "source" (default, the color of the entity with the larger outgoing flow of the pair), "target", or "gradient" (a gradient from one entity's color to the other's).
sortGroups	Order of the arcs around the circle: "none" (default, the order of first appearance in the edge list, the factor levels, or the matrix order), "descending" or "ascending" (by total flow).
padAngle	Gap between adjacent arcs, in radians. Defaults to 0.03.
innerRadiusRatio	Inner radius of the arcs as a fraction of their outer radius (between 0.5 and 0.99). Larger values give thinner arcs. Defaults to 0.92.
arcOpacity	Fill opacity of the arcs (0 to 1). Defaults to 1.
ribbonOpacity	Fill opacity of the ribbons (0 to 1). Defaults to 0.6.
fadeOpacity	Opacity of the ribbons and arcs that are faded out while another element is hovered (0 to 1). Defaults to 0.1.
strokeWidth	Width of the arc and ribbon borders in pixels (the border uses a darker shade of the fill). Defaults to 0.5.
showLabels	Whether to display the entity names outside the arcs. Labels of arcs too small to hold them are hidden. Defaults to TRUE.

rotateLabels	Whether to orient the labels along the radius of the circle (useful with many entities). When FALSE (default), labels are horizontal.
labelFontSize	Font size of the labels. Defaults to 11.
labelOffset	Distance in pixels between the arcs and their labels. Defaults to 10.
valueFormat	A d3-format specifier used to display values in the tooltip (e.g. ",.0f", ".2s", "$,.1f"). Defaults to ",~f".
hover	Whether hovering highlights arcs and ribbons and shows a tooltip, and clicking an arc pins its highlight. Defaults to TRUE.
animate	Whether the arcs sweep in and the ribbons fade in on first render. Defaults to TRUE.
title	Optional. Title of the chart.
titleFontSize	Font size of the chart title. Defaults to 16.
font	The font family used for all text. Defaults to "Verdana, Geneva, Tahoma, sans-serif".
bgcol	Background color of the SVG. Defaults to "white".
textCol	Color of the title and the labels. Defaults to "black".
width	Optional. The width of the SVG output.
height	Optional. The height of the SVG output.

Value

An SVG chord diagram.

Examples

```
# Edge list: hypothetical migration flows between regions (thousands)
migration <- data.frame(
  from  = c("Africa", "Africa", "Africa", "Asia", "Asia", "Asia",
            "Europe", "Europe", "Latin America", "Latin America",
            "North America", "Oceania", "Asia", "Europe"),
  to    = c("Europe", "North America", "Asia", "Europe", "North America",
            "Oceania", "North America", "Asia", "North America", "Europe",
            "Europe", "Asia", "Africa", "Oceania"),
  value = c(420, 150, 90, 610, 520, 230, 180, 120, 870, 260,
            95, 40, 60, 75)
)

chord_diagram(
  data        = migration,
  from        = "from",
  to          = "to",
  value       = "value",
  valueFormat = ",.0f",
  title       = "Migration flows between regions (thousands)"
)

# Square matrix with dimnames: the classic hair-color transition data
```

```
hair <- matrix(
  c(11975,  5871, 8916, 2868,
     1951, 10048, 2060, 6171,
     8010, 16145, 8090, 8045,
     1013,   990,  940, 6907),
  nrow = 4, byrow = TRUE,
  dimnames = list(c("Black", "Blond", "Brown", "Red"),
                  c("Black", "Blond", "Brown", "Red"))
)

chord_diagram(
  data        = hair,
  colors      = c("#3b3b3b", "#f2c45a", "#8c5a3c", "#d0533a"),
  ribbonColor = "gradient",
  padAngle    = 0.05,
  valueFormat = ",.0f",
  title       = "Hair color flows"
)

# Undirected co-authorship links between departments, on a dark background
coauthors <- data.frame(
  from  = c("Biology", "Biology", "Biology", "Chemistry", "Chemistry",
            "Physics", "Physics", "Mathematics", "Computer Science",
            "Computer Science", "Medicine", "Medicine"),
  to    = c("Chemistry", "Medicine", "Computer Science", "Physics",
            "Medicine", "Mathematics", "Computer Science",
            "Computer Science", "Medicine", "Biology", "Chemistry",
            "Physics"),
  value = c(34, 51, 18, 27, 22, 30, 25, 41, 16, 12, 9, 6)
)

chord_diagram(
  data         = coauthors,
  directed     = FALSE,
  sortGroups   = "descending",
  colorPalette = "Set2",
  ribbonOpacity = 0.7,
  bgcol        = "#1e1e2e",
  textCol      = "#f0f0f0",
  title        = "Joint publications between departments"
)
```

`dumbbell_chart` *Create a dumbbell chart.*

Description

A dumbbell chart (also called a connected dot plot) displays two values per category as dots connected by a horizontal line. It is ideal for showing the change or gap between two groups, time points, or conditions across multiple categories — a clean alternative to grouped bar charts favoured by outlets such as the New York Times and The Economist.

Usage

```
dumbbell_chart(
  data,
  x1,
  x2,
  y,
  x1Label = NULL,
  x2Label = NULL,
  col1 = "steelblue",
  col2 = "crimson",
  lineCol = "#aaaaaa",
  lineWidth = 2,
  circleRadius = 6,
  sort = "none",
  xtitle = NULL,
  xtitleFontSize = 13,
  title = NULL,
  titleFontSize = 16,
  xFontSize = 11,
  yFontSize = 11,
  legend = TRUE,
  legendFontSize = 11,
  font = "Verdana, Geneva, Tahoma, sans-serif",
  bgcol = "white",
  axisCol = "black",
  width = NULL,
  height = NULL
)
```

Arguments

`data`	The data frame containing the variables to consider.
`x1`	The name of the column for the first value (left dot).
`x2`	The name of the column for the second value (right dot).
`y`	The name of the categorical column used as row labels.
`x1Label`	Legend label for the first value. Defaults to the value of `x1`.
`x2Label`	Legend label for the second value. Defaults to the value of `x2`.
`col1`	Color of the first dot. Defaults to `"steelblue"`.
`col2`	Color of the second dot. Defaults to `"crimson"`.
`lineCol`	Color of the line connecting the two dots. Defaults to `"#aaaaaa"`.
`lineWidth`	Width of the connecting line in pixels. Defaults to `2`.
`circleRadius`	Radius of the dots in pixels. Defaults to `6`.
`sort`	Whether to sort the rows. One of `"none"` (default), `"ascending"`, or `"descending"`, ordered by the `x1` value.
`xtitle`	Optional. Title of the x-axis.

xtitleFontSize	Font size of the x-axis title. Defaults to 13.
title	Optional. Title of the chart.
titleFontSize	Font size of the chart title. Defaults to 16.
xFontSize	Font size of the x-axis tick labels. Defaults to 11.
yFontSize	Font size of the y-axis category labels. Defaults to 11.
legend	Whether to display a legend. Defaults to TRUE.
legendFontSize	Font size of the legend text. Defaults to 11.
font	The font family used for all text. Defaults to "Verdana, Geneva, Tahoma, sans-serif".
bgcol	Background color of the SVG. Defaults to "white".
axisCol	Color of axis lines, ticks, and labels. Defaults to "black".
width	Optional. The width of the SVG output.
height	Optional. The height of the SVG output.

Value

An SVG dumbbell chart.

Examples

```
# Life expectancy change between 1952 and 2007 for selected countries
life_exp <- data.frame(
  country = c("Brazil", "China", "Egypt", "India",
              "Japan", "Mexico", "Nigeria", "Turkey"),
  year_1952 = c(50.9, 44.0, 41.9, 37.4, 63.0, 50.8, 36.3, 43.6),
  year_2007 = c(72.4, 72.9, 71.3, 64.7, 82.6, 76.2, 46.9, 71.8)
)

dumbbell_chart(
  data     = life_exp,
  x1       = "year_1952",
  x2       = "year_2007",
  y        = "country",
  x1Label  = "1952",
  x2Label  = "2007",
  title    = "Life expectancy: 1952 vs 2007",
  xtitle   = "Life expectancy (years)",
  sort     = "ascending"
)

# Comparing city vs highway fuel economy by car class
library(ggplot2)
library(dplyr)

mpg_summary <- summarise(group_by(mpg, class),
  city = mean(cty), highway = mean(hwy))

dumbbell_chart(
```

```
  data    = mpg_summary,
  x1      = "city",
  x2      = "highway",
  y       = "class",
  x1Label = "City",
  x2Label = "Highway",
  col1    = "steelblue",
  col2    = "darkorange",
  title   = "City vs highway fuel economy by class",
  xtitle  = "Miles per gallon",
  sort    = "ascending"
)
```

`flame` *Display an Animated Flame Visualization*

Description

Creates an animated flame SVG visualization whose size and color gradient can be customized. The flame grows or shrinks based on the intensity parameter, with smooth pulsing and wobbling animation.

Usage

```
flame(
  intensity = 50,
  flameGradientColors = c("white", "yellow", "darkred"),
  flameOutline = "darkred",
  bgcol = "white",
  width = NULL,
  height = NULL
)
```

Arguments

`intensity`	Numeric value controlling the size of the flame. Values greater than 100 cause the flame to grow beyond default scaling, while smaller values shrink it. Defaults to 50.
`flameGradientColors`	A length-3 character vector specifying the colors of the flame gradient, from the center outward. Defaults to `c("white", "yellow", "darkred")`.
`flameOutline`	Color string for the flame's outline stroke. Defaults to "darkred".
`bgcol`	Background color of the SVG canvas. Defaults to "white".
`width`	Optional width of the SVG output.
`height`	Optional height of the SVG output.

Value

An r2d3 object displaying the animated flame visualization.

Examples

```
flame(intensity = 5)
flame(
  intensity = 50,
  flameGradientColors = c("lightblue", "blue", "darkblue")
)
```

---

flower *Display a Rotating Flower Visualization*

Description

Display a Rotating Flower Visualization

Usage

```
flower(
  petalCount = 6,
  petalLength = 100,
  petalWidth = 60,
  petalColor = "lightpink",
  petalStroke = "deeppink",
  centerRadius = 20,
  centerColor = "gold",
  centerStroke = "darkorange",
  centerText = NULL,
  centerTextSize = 16,
  centerTextColor = "black",
  font = "Verdana, Geneva, Tahoma, sans-serif",
  bgcol = "white",
  rotationSpeed = 2,
  width = NULL,
  height = NULL
)
```

Arguments

petalCount	The number of petals. Defaults to 6.
petalLength	The length of each petal. Defaults to 100.
petalWidth	The width of each petal. Defaults to 60.
petalColor	The fill color of the petals. Defaults to "lightpink".
petalStroke	The stroke color of the petals. Defaults to "deeppink".

centerRadius	The radius of the flower's center circle. Defaults to 20.
centerColor	The fill color of the center. Defaults to "gold".
centerStroke	The stroke color of the center. Defaults to "darkorange".
centerText	Optional. Text to display inside the center (e.g., a number or emoji).
centerTextSize	The size of the center text. Defaults to 16.
centerTextColor	The color of the center text. Defaults to "black".
font	The font family for the center text. Defaults to "Verdana, Geneva, Tahoma, sans-serif".
bgcol	The background color of the visualization. Defaults to "white".
rotationSpeed	The speed of rotation (degrees per animation frame). Defaults to 2.
width	The width of the SVG output. Optional.
height	The height of the SVG output. Optional.

Value

An animated rotating flower SVG.

Examples

```
flower(
  petalCount = 5,
  petalColor = "plum",
  rotationSpeed = 1.5
)
```

---

gauge_chart *Create a gauge (speedometer) chart.*

Description

A gauge (speedometer) chart displays a single quantitative measure against colored zones (warning and danger) with a needle indicator. By default the needle animates on load, sweeping up from the minimum to the value and settling with a small wobble, like a car dashboard.

Usage

```
gauge_chart(
  value,
  min = 0,
  max = 100,
  title = "",
  warningZone = NULL,
  warningColor = "orange",
  dangerZone = NULL,
```

```
  dangerColor = "red",
  arcBgColor = "#ddd",
  needleColor = "crimson",
  pivotColor = "steelblue",
  titleFontSize = 16,
  valueFontSize = 14,
  font = "Verdana, Geneva, Tahoma, sans-serif",
  bgcol = "white",
  animate = TRUE,
  animationDuration = 1500,
  width = NULL,
  height = NULL
)
```

Arguments

value	Numeric. The current value to display on the gauge.
min	Numeric. The minimum value of the gauge. Defaults to 0.
max	Numeric. The maximum value of the gauge. Defaults to 100.
title	Character. Label shown inside the gauge. Defaults to "".
warningZone	Numeric. The value at which the warning (first) arc begins. Defaults to 75% of max.
warningColor	The color of the warning zone arc. Defaults to "orange".
dangerZone	Numeric. The value at which the danger (second, critical) arc begins. Defaults to 90% of max.
dangerColor	The color of the danger zone arc. Defaults to "red".
arcBgColor	The background color of the gauge arc track. Defaults to "#ddd".
needleColor	The color of the needle. Defaults to "crimson".
pivotColor	The color of the center pivot circle. Defaults to "steelblue".
titleFontSize	The font size of the gauge title. Defaults to 16.
valueFontSize	The font size of the current value label. Defaults to 14.
font	The font family used for all text. Defaults to "Verdana, Geneva, Tahoma, sans-serif".
bgcol	The background color of the SVG. Defaults to "white".
animate	Logical. When TRUE (default), the needle sweeps from the minimum up to the value on load and settles with a small wobble. Set to FALSE to draw the needle at the value immediately with no animation.
animationDuration	Numeric. Duration of the needle sweep in milliseconds. Defaults to 1500.
width	Optional. The width of the SVG output.
height	Optional. The height of the SVG output.

Value

An SVG gauge chart.

Examples

```
# Basic usage
gauge_chart(value = 8, title = "Memory")

# Custom warning and danger thresholds
gauge_chart(
  value = 72,
  min = 0,
  max = 100,
  title = "CPU Load",
  warningZone  = 60,
  warningColor = "orange",
  dangerZone   = 80,
  dangerColor  = "red"
)

# Non-percentage range (response time in ms)
gauge_chart(
  value = 530,
  min = 0,
  max = 1000,
  title = "Response (ms)",
  warningZone  = 400,
  dangerZone   = 700
)
```

---

glass_fill *Visualize a Glass Filling with Water using D3*

Description

This function generates an SVG visualization of a glass filled with water to a specified level, rendered via D3 using the r2d3 package. The fill level, appearance of the glass, and label settings can be customized.

Usage

```
glass_fill(
  fill_level = 0.65,
  glassWidth = 80,
  glassHeight = 200,
  strokeColor = "#555",
  strokeWidth = 3,
  fillColor = "skyblue",
  renderFillLabel = TRUE,
  labelFontSize = "16px",
  titleText = "Fill level",
  labelColor = "#333",
```

```
  titleColor = "#333",
  titleFontSize = "14px",
  font = "Verdana, Geneva, Tahoma, sans-serif"
)
```

Arguments

fill_level	Numeric value between 0 and 1 indicating how full the glass should appear.
glassWidth	Width of the glass in pixels.
glassHeight	Height of the glass in pixels.
strokeColor	Color of the glass outline (stroke).
strokeWidth	Width of the glass outline stroke.
fillColor	Color used to fill the water in the glass.
renderFillLabel	Logical indicating whether to display a percentage label above the glass.
labelFontSize	Font size of the label, defaults to "16px"
titleText	Text to display as the title beneath the glass.
labelColor	Color of the label.
titleColor	Color of the title text displayed below the glass.
titleFontSize	Font size of the title text, defaults to "14px"
font	The font name that will be used for the plot text. Defaults to "Verdana, Geneva, Tahoma, sans-serif"

Value

An interactive D3 visualization rendered in the RStudio Viewer or web browser.

Examples

```
glass_fill(fill_level = 0.75)
glass_fill(fill_level = 0.3, fillColor = "lightblue", titleText = "Water Intake")
```

---

heart_fill *Visualize a Heart Filling with Color using D3*

Description

This function renders a heart-shaped SVG graphic that fills from the bottom up based on the provided level. The appearance of the heart and the optional label can be fully customized. It uses the ‘r2d3‘ package to render the visualization with D3.js.

Usage

```
heart_fill(
  fill_level = 0.65,
  heartSize = 150,
  strokeColor = "#C00",
  strokeWidth = 4,
  fillColor = "red",
  renderFillLabel = TRUE,
  labelColor = "#333",
  labelFontSize = "16px",
  titleText = "Fill level",
  titleColor = "#333",
  titleFontSize = "14px",
  font = "Verdana, Geneva, Tahoma, sans-serif"
)
```

Arguments

`fill_level`	Value between 0 and 1 indicating how full the heart should appear (e.g., 0.65).
`heartSize`	Width/height scale of the heart in pixels.
`strokeColor`	Color of the heart outline.
`strokeWidth`	Width of the heart outline stroke.
`fillColor`	Color used to fill the heart based on the fill level.
`renderFillLabel`	Whether to display a percentage label above the heart.
`labelColor`	Color of the percentage label text.
`labelFontSize`	Font size of the percentage label text (e.g., "16px").
`titleText`	Optional title displayed below the heart.
`titleColor`	Color of the title text.
`titleFontSize`	Font size of the title text (e.g., "14px").
`font`	Font family used for text labels and title.

Value

An interactive D3 heart fill visualization rendered in the RStudio Viewer or web browser.

Examples

```
heart_fill(fill_level = 0.9)
heart_fill(
  fill_level = 0.4,
  fillColor = "pink",
  labelColor = "#C00",
  strokeColor = "#900"
)
```

hexbin_plot *Create a hexbin plot.*

Description

A hexbin plot is a two-dimensional histogram: the x/y plane is tiled with a grid of regular hexagons, every observation is assigned to the hexagon it falls into, and each hexagon is colored by the number of observations it contains. It is a readable alternative to a scatter plot when there are too many points to see individually (overplotting), because dense regions remain distinguishable instead of turning into a solid blob.

The look is inspired by the `hexbin` transform of the Observable Plot library: light gridlines, a sequential color scale with a continuous legend, and an optional mode where the hexagon *size* (area) also encodes the count. Binning is done in pixel space, so the hexagons are always regular regardless of the units of `x` and `y`.

The plot is interactive: hovering a hexagon outlines it, marks its count on the color legend and (when `tooltip = TRUE`) shows a tooltip with the bin count, its share of all observations, and the x/y range covered by the hexagon.

Usage

```
hexbin_plot(
  data,
  x,
  y,
  binWidth = 12,
  sizeByCount = FALSE,
  colorPalette = "YlGnBu",
  reverseColors = FALSE,
  colorScale = c("linear", "sqrt", "log"),
  legend = TRUE,
  legendTitle = "Count",
  tooltip = TRUE,
  hover = TRUE,
  animate = TRUE,
  opacity = 1,
  stroke = "white",
  strokeWidth = 0.5,
  grid = TRUE,
  gridCol = "#e5e5e5",
  xtitle = NULL,
  ytitle = NULL,
  xtitleFontSize = 13,
  ytitleFontSize = 13,
  title = NULL,
  titleFontSize = 16,
  xFontSize = 11,
```

```
  yFontSize = 11,
  font = "Verdana, Geneva, Tahoma, sans-serif",
  bgcol = "white",
  axisCol = "black",
  width = NULL,
  height = NULL
)
```

Arguments

data	The data frame containing the variables to consider.
x	The name of the numeric column mapped to the horizontal axis.
y	The name of the numeric column mapped to the vertical axis.
binWidth	The radius of each hexagon in pixels (distance from its center to a corner). Larger values give fewer, coarser bins. Defaults to 12.
sizeByCount	Logical. When TRUE, the area of each hexagon is also proportional to its count (the largest bin fills its cell), in addition to the fill color. Defaults to FALSE.
colorPalette	The name of a D3 sequential color interpolator, without the "interpolate" prefix (e.g. "Viridis", "Blues", "YlGnBu", "Magma", "Turbo"). Defaults to "YlGnBu".
reverseColors	Logical. Whether to reverse the direction of the color palette. Defaults to FALSE.
colorScale	How counts are mapped to colors: "linear" (default), "sqrt" or "log". Use "sqrt" or "log" when a few very dense bins wash out the rest.
legend	Logical. Whether to draw the continuous color legend. Defaults to TRUE.
legendTitle	Title of the color legend. Defaults to "Count".
tooltip	Logical. Whether to show a tooltip when a hexagon is hovered. Defaults to TRUE.
hover	Logical. When TRUE (default), hovering a hexagon outlines it and marks its count on the legend.
animate	Logical. When TRUE (default), hexagons grow into place with a short staggered entry animation on first render.
opacity	Fill opacity of the hexagons (0 to 1). Defaults to 1.
stroke	Stroke color of the hexagons. Defaults to "white".
strokeWidth	Stroke width of the hexagons in pixels. Defaults to 0.5.
grid	Logical. Whether to draw light gridlines behind the hexagons. Defaults to TRUE.
gridCol	Color of the gridlines. Defaults to "#e5e5e5".
xtitle	Optional. Title of the x-axis. Defaults to the value of x.
ytitle	Optional. Title of the y-axis. Defaults to the value of y.
xtitleFontSize	Font size of the x-axis title. Defaults to 13.
ytitleFontSize	Font size of the y-axis title. Defaults to 13.
title	Optional. Title of the chart.
titleFontSize	Font size of the chart title. Defaults to 16.

xFontSize	Font size of the x-axis tick labels. Defaults to 11.
yFontSize	Font size of the y-axis tick labels. Defaults to 11.
font	The font family used for all text. Defaults to "Verdana, Geneva, Tahoma, sans-serif".
bgcol	Background color of the SVG. Defaults to "white".
axisCol	Color of axis lines, ticks, labels and titles. Defaults to "black".
width	Optional. The width of the SVG output.
height	Optional. The height of the SVG output.

Value

An SVG hexbin plot.

Examples

```
# Old Faithful geyser: eruption duration vs waiting time
hexbin_plot(
  data     = faithful,
  x        = "waiting",
  y        = "eruptions",
  binWidth = 14,
  xtitle   = "Waiting time to next eruption (min)",
  ytitle   = "Eruption duration (min)",
  title    = "Old Faithful eruptions"
)

# 10,000 diamonds: carat vs price, with a log color scale
library(ggplot2) # for the diamonds dataset
set.seed(1)
dsample <- diamonds[sample(nrow(diamonds), 10000), ]

hexbin_plot(
  data         = dsample,
  x            = "carat",
  y            = "price",
  binWidth     = 8,
  colorPalette = "Viridis",
  colorScale   = "log",
  xtitle       = "Carat",
  ytitle       = "Price (USD)",
  title        = "Diamond price by carat"
)

# Hexagon size encodes the count as well as the color
set.seed(42)
pts <- data.frame(
  a = c(rnorm(1500, 0, 1), rnorm(800, 3, 0.6)),
  b = c(rnorm(1500, 0, 1), rnorm(800, 2, 0.6))
)
```

```
hexbin_plot(
  data         = pts,
  x            = "a",
  y            = "b",
  binWidth     = 13,
  sizeByCount  = TRUE,
  colorPalette = "Magma",
  reverseColors = TRUE,
  title        = "Two bivariate normal clusters"
)
```

`histogram` *Create a histogram.*

Description

Create a histogram.

Usage

```
histogram(
  x,
  bins = 30,
  fill = "crimson",
  xFontSize = 10,
  yFontSize = 10,
  xticks = NULL,
  yticks = NULL,
  xtitle = NULL,
  xtitleFontSize = 16,
  ytitle = NULL,
  ytitleFontSize = 16,
  title = NULL,
  titleFontSize = 22,
  stroke = "crimson",
  strokeWidth = NULL,
  font = "Verdana, Geneva, Tahoma, sans-serif",
  bgcol = "#CAD0D3",
  opacity = 1,
  axisCol = "black",
  width = NULL,
  height = NULL
)
```

Arguments

`x`	A vector of data.
`bins`	The number of bins to consider. Defaults to 30.

fill	The color of the bars. Defaults to 'crimson'.
xFontSize	the font size of the x-axis labels. Defaults to 10.
yFontSize	the font size of the y-axis labels. Defaults to 10.
xticks	Optional. the number of x-axis ticks to consider.
yticks	Optional. The number of y-axis ticks to consider.
xtitle	Optional. The title of the x-axis.
xtitleFontSize	The font size of the x-axis title. Defaults to 16.
ytitle	Optional. The title of the y-axis.
ytitleFontSize	The font size of the y-axis title. Defaults to 16.
title	Optional. The title of the plot.
titleFontSize	The font size of the plot title. Defaults to 22.
stroke	The stroke color of the bars. Defaults to 'crimson'.
strokeWidth	Optional. the stroke width of the bars.
font	The font family to consider for the titles. Defaults to "Verdana, Geneva, Tahoma, sans-serif".
bgcol	The background color of the SVG. Defaults to "#CAD0D3" HEX color.
opacity	The color opacity of the bars (from 0 to 1). Defaults to 1.
axisCol	the color of the x and y axis. It includes the ticks, the labels and titles. Defaults to 'black'.
width	Optional. The width of the SVG output.
height	Optional. The height of the SVG output.

Value

A SVG histogram.

Examples

```
histogram(
  x = mtcars$mpg,
  bins = 20,
  fill = "crimson",
  stroke = "white",
  strokeWidth = 1,
  title = "Distribution of the hwy variable",
  width = "20",
  height = "10"
)
```

`horz_bar_chart` *Create a horizontal bar chart*

Description

Create a horizontal bar chart

Usage

```
horz_bar_chart(
  data,
  label,
  value,
  fill = "crimson",
  sort = "none",
  paddingWidth = 0.1,
  stroke = NULL,
  strokeWidth = 1,
  bgcol = "#CAD0D3",
  valueTicks = NULL,
  valueFontSize = 10,
  labelFontSize = 10,
  valueTitle = NULL,
  valueTitleFontSize = 14,
  labelTitle = NULL,
  labelTitleFontSize = 14,
  font = "Verdana, Geneva, Tahoma, sans-serif",
  title = NULL,
  titleFontSize = 20,
  opacity = 1,
  axisCol = "black",
  width = NULL,
  height = NULL
)
```

Arguments

`data`	The data frame containing the variables to consider.
`label`	The categorical variable to consider. Will be plotted on the y-axis.
`value`	The numeric variable to consider. Will be plotted on the x-axis.
`fill`	The color of the bars. Defaults to 'crimson'.
`sort`	Optional. Takes the following arguments: 'none', 'ascending' or 'descending', default to 'none'
`paddingWidth`	The distance between each bar. The value goes from 0 to 0.99 included. Defaults to 0.1.

stroke	Optional. The color of the stroke of the bars.
strokeWidth	The width of the stroke of the bars. Defaults to 1 when the 'stroke' parameter is used.
bgcol	Optional. The color of the background, default to: '#CAD0D3'
valueTicks	Optional. the number of x-axis ticks to consider.
valueFontSize	The font size of the x-axis values. Defaults to 10.
labelFontSize	The font size of the y-axis labels. Defaults to 10.
valueTitle	Optional. The title of the x-axis.
valueTitleFontSize	The font size of the x-axis title if specified. Defaults to 14.
labelTitle	Optional. The title of the y-axis.
labelTitleFontSize	The font size of the y-axis title. Defaults to 14.
font	The font family of the text. Defaults to "Verdana, Geneva, Tahoma, sans-serif"
title	Optional. The title of the overall graphic.
titleFontSize	The font size of the overall graphic's title when specified.
opacity	The color opacity of the bars. Goes from 0 to 1. Defaults to 1.
axisCol	the color of the x and y axis. It includes the ticks, the labels and titles. Defaults to 'black'.
width	Optional. The width of the SVG output.
height	Optional. The height of the SVG output.

Value

A SVG horizontal bar chart.

Examples

```
library(ggplot2) # needed for the mpg data frame
library(dplyr) # needed for the data wrangling process

mpg %>% group_by(manufacturer) %>%
 summarise(median_hwy = median(hwy)) %>%
 horz_bar_chart(
   label = "manufacturer",
   value = "median_hwy",
   sort = "ascending"
 )
```

horz_lollipop *Create a horizontal lollipop chart*

Description

Create a horizontal lollipop chart

Usage

```
horz_lollipop(
  data,
  label,
  value,
  sort = "none",
  bgcol = "white",
  valueTicks = NULL,
  labelTicks = NULL,
  valueFontSize = 12,
  labelFontSize = 12,
  font = "Verdana, Geneva, Tahoma, sans-serif",
  valueTitle = NULL,
  valueTitleFontSize = 14,
  labelTitle = NULL,
  labelTitleFontSize = 14,
  title = NULL,
  titleFontSize = 20,
  lineStroke = "maroon",
  lineStrokeWidth = 4,
  circleFill = "lime",
  circleStroke = "lime",
  circleStrokeWidth = 1,
  circleRadius = 5,
  axisCol = "black",
  width = NULL,
  height = NULL
)
```

Arguments

data	The data frame containing the variables to consider.
label	The categorical variable to consider. Will be plotted on the x-axis.
value	The numeric variable to consider. Will be plotted on the y-axis.
sort	Whether to sort or not the vertical lines. Takes three values 'none' which is the default, 'ascending' or 'descending'.
bgcol	The background-color of the SVG output. Defaults to 'salmon'.

valueTicks	Optional. the number of x-axis ticks to consider.
labelTicks	Optional. The number of y-axis ticks to consider.
valueFontSize	the font size of the x-axis labels. Defaults to 10.
labelFontSize	the font size of the y-axis labels. Defaults to 10.
font	The font family to consider for the titles. Defaults to "Verdana, Geneva, Tahoma, sans-serif".
valueTitle	Optional. The title of the x-axis.
valueTitleFontSize	The font size of the x-axis title. Defaults to 14.
labelTitle	Optional. The title of the y-axis.
labelTitleFontSize	The font size of the y-axis title. Defaults to 14.
title	Optional. The title of the plot.
titleFontSize	The font size of the plot title. Defaults to 22.
lineStroke	The stroke color of the vertical lines. Defaults to 'maroon'.
lineStrokeWidth	The vertical lines stroke's width. Defaults to 4.
circleFill	The color of the circles. Defaults to 'lime'.
circleStroke	The color of the stroke surrounding the circle. Defaults to 'lime'.
circleStrokeWidth	The width of the circles' stroke. Defaults to 1.
circleRadius	The radius of the circles. Defaults to 10.
axisCol	the color of the x and y axis. It includes the ticks, the labels and titles. Defaults to 'black'.
width	Optional. The width of the SVG output.
height	Optional. The height of the SVG output.

Value

A SVG horizontal lollipop chart.

`line_chart` *Create an interactive time series line chart*

Description

Draws one or several time series as lines, with the interactions that people who work with time series expect: a crosshair tooltip that shows every series at the hovered date, a clickable legend, direct labels at the end of each line, range buttons (1M, 6M, YTD, 1Y, ...), a navigator with a brush under the chart, drag-to-zoom and double-click to reset. The y axis rescales to the data in view whenever you zoom.

Usage

```
line_chart(
  data,
  x = NULL,
  y = NULL,
  group = NULL,
  lower = NULL,
  upper = NULL,
  colors = NULL,
  stroke = NULL,
  strokeWidth = 2,
  curve = "curveLinear",
  opacity = 1,
  area = FALSE,
  points = FALSE,
  yScale = c("linear", "log"),
  yZero = NULL,
  transform = c("none", "index", "percent"),
  indexBase = 100,
  rollmean = NULL,
  showRaw = TRUE,
  events = NULL,
  bands = NULL,
  hlines = NULL,
  legend = NULL,
  endLabels = NULL,
  tooltip = TRUE,
  navigator = TRUE,
  rangeButtons = TRUE,
  zoom = TRUE,
  animate = TRUE,
  valueFormat = NULL,
  valuePrefix = "",
  valueSuffix = "",
  dateFormat = NULL,
  xticks = NULL,
  yticks = NULL,
  xtitle = NULL,
  xtitleFontSize = 13,
  ytitle = NULL,
  ytitleFontSize = 13,
  title = NULL,
  titleFontSize = 20,
  subtitle = NULL,
  subtitleFontSize = 13,
  xFontSize = 11,
  yFontSize = 11,
  font = "Verdana, Geneva, Tahoma, sans-serif",
```

```
  bgcol = "white",
  gridCol = "#e9e9e9",
  axisCol = "#333333",
  width = NULL,
  height = NULL
)
```

Arguments

data	A data frame, or a ts / mts object.
x	The time variable. Can be a Date, a POSIXct date-time, a character column in "yyyy-mm-dd" format, or a numeric column (for example years). Not needed when data is a ts object.
y	The value variable(s). One column name, or several column names to draw one line per column (wide data). For a univariate ts object, it is the name given to the series (defaults to "value").
group	Optional. For long data, the column whose values name the series (one line per value). Cannot be combined with several y columns.
lower, upper	Optional. Columns holding the lower and upper bounds of a confidence or forecast interval, drawn as a shaded ribbon around the line. With several y columns, give one column per y column (use NA for series without a ribbon).
colors	Optional. Colors of the series. Either an unnamed vector (recycled in series order) or a vector named by series. Defaults to a built-in 10-color palette.
stroke	The color of the line when there is a single series. When NULL (default), the first palette color is used.
strokeWidth	The width of the lines. Defaults to 2.
curve	The line's curve type. A complete list can be found here <https://github.com/d3/d3-shape#curves>. Defaults to 'curveLinear'; 'curveMonotoneX' gives smooth lines that never overshoot the data.
opacity	The opacity of the lines (from 0 to 1). Defaults to 1.
area	Logical. Fill the area under each line with a soft gradient. Defaults to FALSE.
points	Logical. Draw a marker on every data point. Defaults to FALSE.
yScale	Either "linear" (default) or "log".
yZero	Logical. Force the y axis to include 0. When NULL (default), 0 is included only when area = TRUE; otherwise the axis fits the data in view.
transform	How values are displayed: "none" (default), "index" (each series rebased to indexBase at the first visible date, so series on different scales can be compared) or "percent" (change since the first visible date). The base moves with the zoom.
indexBase	The value series are rebased to when transform = "index". Defaults to 100.
rollmean	Optional. A window size (number of observations). When given, a trailing moving average is drawn as the main line.
showRaw	Logical. When rollmean is used, also draw the raw series as a thin faded line. Defaults to TRUE.

`events` Optional. Dates to mark with a vertical line: either a vector of dates (names are used as labels) or a data frame whose first column holds the dates and whose second column (optional) holds labels.

`bands` Optional. Periods to shade, as a data frame whose first two columns are the start and end dates. An optional third column holds the labels and an optional fourth column the fill colors.

`hlines` Optional. Numeric reference values drawn as horizontal dashed lines (names are used as labels). Only drawn when `transform = "none"`.

`legend` Logical. Show the interactive legend (click to hide or show a series, double-click to isolate it). When `NULL` (default), the legend is shown when there are two series or more.

`endLabels` Logical. Label each line at its last visible point. When `NULL` (default), labels are shown when there are two series or more.

`tooltip` Logical. Show the crosshair and the tooltip on hover. Defaults to `TRUE`.

`navigator` Logical. Show the navigator (a miniature of the full series with a brush to select the visible range) under the chart. Defaults to `TRUE`; it is hidden automatically on very short charts.

`rangeButtons` Logical. Show range buttons (1M, 6M, YTD, 1Y, 5Y, All, ...), chosen from the length and frequency of the data. Only used with dates. Defaults to `TRUE`.

`zoom` Logical. Drag horizontally on the chart to zoom into a range; double-click to reset. Defaults to `TRUE`.

`animate` Logical. Draw the lines in with an animation on first render. Defaults to `TRUE`.

`valueFormat` Optional. A d3-format string for values in the tooltip (see <https://github.com/d3/d3-format>). Defaults to ",.2~f", or "+.2~%" with `transform = "percent"`.

`valuePrefix, valueSuffix`
Optional. Text placed before and after the values on the y axis and in the tooltip, for example "$" or " km".

`dateFormat` Optional. A d3-time-format string for the date in the tooltip (see <https://github.com/d3/d3-time-format>). When `NULL` (default), it is picked from the frequency of the data (yearly, quarterly, monthly, daily or intraday).

`xticks` Optional. The approximate number of x-axis ticks.

`yticks` Optional. The approximate number of y-axis ticks.

`xtitle` Optional. The title of the x-axis.

`xtitleFontSize` The font size of the x-axis title. Defaults to 13.

`ytitle` Optional. The title of the y-axis, shown above the axis.

`ytitleFontSize` The font size of the y-axis title. Defaults to 13.

`title` Optional. The title of the plot.

`titleFontSize` The font size of the plot title. Defaults to 20.

`subtitle` Optional. A subtitle shown under the title.

`subtitleFontSize`
The font size of the subtitle. Defaults to 13.

`xFontSize`, `yFontSize`
: The font size of the axis tick labels. Defaults to 11.

`font`
: The font family to consider for the titles. Defaults to "Verdana, Geneva, Tahoma, sans-serif".

`bgcol`
: The background color of the SVG. Defaults to "white".

`gridCol`
: The color of the horizontal grid lines. Defaults to "#e9e9e9".

`axisCol`
: The color of the axis text, ticks and titles. Defaults to "#333333".

`width`
: Optional. The width of the SVG output.

`height`
: Optional. The height of the SVG output.

Details

Series can be given in three ways:

- **one column**: `y = "value"`;
- **wide data**: several columns, `y = c("a", "b", "c")`;
- **long data**: one value column and a `group` column that names the series.

A `ts` or `mts` object (for example `AirPassengers` or `EuStockMarkets`) can also be passed directly as `data`, in which case `x` and `y` are not needed. Monthly, quarterly and yearly series get exact calendar dates; for other frequencies (for example the 260 trading days a year of `EuStockMarkets`) the dates are interpolated from the decimal time index, so pass a data frame with a real date column when exact dates matter.

Time-series extras: `transform = "index"` rebases every series to 100 at the start of the visible range (or `"percent"` for the change since then), `rollmean` overlays a moving average, `lower` / `upper` draw a confidence or forecast ribbon, `bands` shade periods such as recessions, `events` mark dates with a vertical line and `hlines` draw reference values.

Missing values (`NA`) leave a gap in the line.

Value

An interactive SVG line chart.

Examples

```
# A ts object can be passed directly
line_chart(AirPassengers, y = "passengers", title = "Airline passengers")

# The classic data frame interface still works
airpassengers <- data.frame(
  passengers = as.matrix(AirPassengers),
  date = zoo::as.Date(time(AirPassengers))
)
line_chart(data = airpassengers, x = "date", y = "passengers")

# Several series from an mts object, rebased to 100 at the start of the
# visible range: zoom with the range buttons or the navigator and the
# comparison is recomputed
```

```
line_chart(
  EuStockMarkets,
  transform = "index",
  title = "European stock indices",
  subtitle = "Daily closing prices, rebased to 100"
)

# Long data with a group column, area fill and shaded recessions
library(ggplot2) # for the economics data set

recessions <- data.frame(
  start = as.Date(c("1969-12-01", "1973-11-01", "1980-01-01", "1981-07-01",
                    "1990-07-01", "2001-03-01", "2007-12-01")),
  end   = as.Date(c("1970-11-01", "1975-03-01", "1980-07-01", "1982-11-01",
                    "1991-03-01", "2001-11-01", "2009-06-01"))
)

line_chart(
  economics,
  x = "date",
  y = "unemploy",
  area = TRUE,
  bands = recessions,
  valueFormat = ",.0f",
  valueSuffix = "k",
  title = "US unemployment",
  subtitle = "Thousands of persons, recessions shaded"
)

# A forecast with its prediction interval
fit <- HoltWinters(log(AirPassengers))
fc <- predict(fit, n.ahead = 24, prediction.interval = TRUE)
fc_dates <- zoo::as.Date(time(fc))

passengers <- rbind(
  data.frame(date = zoo::as.Date(time(AirPassengers)),
             value = as.numeric(AirPassengers), lo = NA, hi = NA,
             series = "Observed"),
  data.frame(date = fc_dates, value = exp(fc[, "fit"]),
             lo = exp(fc[, "lwr"]), hi = exp(fc[, "upr"]),
             series = "Forecast")
)

line_chart(
  passengers,
  x = "date", y = "value", group = "series",
  lower = "lo", upper = "hi",
  colors = c(Observed = "#4269d0", Forecast = "#ff725c"),
  events = c("Forecast starts" = as.Date("1961-01-01")),
  valueFormat = ",.0f",
  title = "Airline passengers: Holt-Winters forecast"
)
```

liquid_chart *Create a liquid fill gauge chart.*

Description

A liquid fill gauge represents a single value as a circle filled with an animated liquid up to the corresponding level. The fill rises smoothly on load, two wave layers give the water a natural depth, and the label text changes color at the water surface — appearing in `textColor` above the water and in `waveTextColor` below it.

Usage

```
liquid_chart(
  value,
  label = NULL,
  fillColor = "steelblue",
  circleColor = NULL,
  circleThickness = 0.05,
  textColor = NULL,
  waveTextColor = "white",
  textSize = 0.38,
  waveAmplitude = 0.09,
  waveCount = 2,
  waveSpeed = 1,
  title = NULL,
  titleFontSize = 14,
  font = "Verdana, Geneva, Tahoma, sans-serif",
  bgcol = "white",
  width = NULL,
  height = NULL
)
```

Arguments

`value`	Numeric. The fill level, between 0 and 1 (0 = empty, 1 = full).
`label`	Character. Text displayed in the center of the gauge. Defaults to `value` formatted as a percentage (e.g. `"55%"`).
`fillColor`	The color of the liquid. Defaults to `"steelblue"`.
`circleColor`	The color of the outer ring. Defaults to `fillColor`.
`circleThickness`	Thickness of the outer ring as a fraction of the circle radius. Defaults to `0.05`.
`textColor`	Color of the label text above the water surface. Defaults to `fillColor`.
`waveTextColor`	Color of the label text below (inside) the water surface. Defaults to `"white"`.
`textSize`	Font size of the label as a fraction of the circle radius. Defaults to `0.38`.
`waveAmplitude`	Height of the wave as a fraction of the circle radius. Defaults to `0.09`.

waveCount	Number of full wave cycles visible across the circle. Defaults to 2.
waveSpeed	Speed of the horizontal wave animation. Higher values are faster. Defaults to 1.
title	Optional. A label displayed below the circle.
titleFontSize	Font size of the title in pixels. Defaults to 14.
font	The font family used for all text. Defaults to "Verdana, Geneva, Tahoma, sans-serif".
bgcol	Background color of the SVG. Defaults to "white".
width	Optional. The width of the SVG output.
height	Optional. The height of the SVG output.

Value

An animated SVG liquid fill gauge.

Examples

```
# Basic usage — 55% fill
liquid_chart(value = 0.55)

# Custom color and label
liquid_chart(
  value     = 0.72,
  label     = "72%",
  fillColor = "#e67e22",
  title     = "Disk usage"
)

# Low fill with a raw value label
liquid_chart(
  value         = 0.28,
  label         = "28%",
  fillColor     = "tomato",
  circleColor   = "tomato",
  textColor     = "tomato",
  waveTextColor = "white"
)
```

---

lollipop_chart *Create a lollipop chart*

Description

Create a lollipop chart

Usage

```
lollipop_chart(
  data,
  x,
  y,
  sort = "none",
  bgcol = "white",
  xticks = NULL,
  yticks = NULL,
  xFontSize = 12,
  yFontSize = 12,
  font = "Verdana, Geneva, Tahoma, sans-serif",
  xtitle = NULL,
  xtitleFontSize = 14,
  ytitle = NULL,
  ytitleFontSize = 14,
  title = NULL,
  titleFontSize = 20,
  lineStroke = "maroon",
  lineStrokeWidth = 4,
  circleFill = "lime",
  circleStroke = "lime",
  circleStrokeWidth = 1,
  circleRadius = 10,
  axisCol = "black",
  width = NULL,
  height = NULL
)
```

Arguments

data	The data frame containing the variables to consider.
x	The categorical variable to consider. Will be plotted on the x-axis.
y	The numeric variable to consider. Will be plotted on the y-axis.
sort	Whether to sort or not the vertical lines. Takes three values 'none' which is the default, 'ascending' or 'descending'.
bgcol	The background-color of the SVG output. Defaults to 'white'.
xticks	Optional. the number of x-axis ticks to consider.
yticks	Optional. The number of y-axis ticks to consider.
xFontSize	the font size of the x-axis labels. Defaults to 10.
yFontSize	the font size of the y-axis labels. Defaults to 10.
font	The font family to consider for the titles. Defaults to "Verdana, Geneva, Tahoma, sans-serif".
xtitle	Optional. The title of the x-axis.
xtitleFontSize	The font size of the x-axis title. Defaults to 16.

ytitle	Optional. The title of the y-axis.
ytitleFontSize	The font size of the y-axis title. Defaults to 16.
title	Optional. The title of the plot.
titleFontSize	The font size of the plot title. Defaults to 22.
lineStroke	The stroke color of the vertical lines. Defaults to 'maroon'.
lineStrokeWidth	The vertical lines stroke's width. Defaults to 4.
circleFill	The color of the circles. Defaults to 'lime'.
circleStroke	The color of the stroke surrounding the circle. Defaults to 'lime'.
circleStrokeWidth	The width of the circles' stroke. Defaults to 1.
circleRadius	The radius of the circles. Defaults to 10.
axisCol	the color of the x and y axis. It includes the ticks, the labels and titles. Defaults to 'black'.
width	Optional. The width of the SVG output.
height	Optional. The height of the SVG output.

Value

A SVG lollipop chart.

Examples

```
library(ggplot2) # needed for the mpg data frame
library(dplyr) # needed for data wrangling

mpg %>% group_by(drv) %>%
  summarise(median_cty = median(cty)) %>%
  lollipop_chart(
    x = "drv",
    y = "median_cty",
    sort = "ascending",
    xtitle = "drv variable",
    ytitle = "median cty",
    title = "Median cty per drv"
  )
```

---

mosaic_chart *Create a mosaic (Marimekko) chart.*

Description

A mosaic chart, also called a Marimekko or Mekko chart, shows how two categorical variables relate to each other and to the whole. Each column stands for one category of `x`, and its width is that category's share of the grand total. Each column is then split into stacked segments, one per category of `fill`, normalised to 100% of the column. So the area of every cell is proportional to its count (or summed value), and you can read both marginal and conditional shares in a single picture. The look follows the mosaic example from the Vega-Lite gallery: white gaps between cells, percentage axes, column labels across the top and in-cell labels wherever they fit.

Hovering a cell highlights it and shows a tooltip with its count, the share of its column in the total and its share within the column. Hovering a legend entry highlights every cell of that `fill` category, and hovering a column label highlights the whole column.

Column and segment order follow the factor levels of `x` and `fill`. Character columns are ordered alphabetically and numeric columns in increasing order.

Usage

```
mosaic_chart(
  data,
  x,
  fill,
  value = NULL,
  colors = NULL,
  gap = 2,
  labelType = c("percent", "count", "both", "none"),
  valueFormat = ",.0f",
  labelFontSize = 11,
  labelCol = NULL,
  columnFontSize = 12,
  hover = TRUE,
  animate = TRUE,
  legend = TRUE,
  legendTitle = NULL,
  legendFontSize = 11,
  xtitle = NULL,
  ytitle = NULL,
  axisTitleFontSize = 12,
  xFontSize = 10,
  yFontSize = 10,
  title = NULL,
  titleFontSize = 16,
  font = "Verdana, Geneva, Tahoma, sans-serif",
  bgcol = "white",
  axisCol = "black",
  width = NULL,
  height = NULL
)
```

Arguments

data	The data frame containing the variables to consider.
x	The name of the categorical column that defines the columns. The width of each column is its share of the total.
fill	The name of the categorical column that splits each column into stacked segments, normalised to 100% within the column.
value	Optional. The name of a numeric (non-negative) column that is summed for each x/fill combination. Defaults to NULL, which counts rows instead.
colors	Optional. A vector of colors, one per fill category, recycled if needed. Defaults to NULL, which uses D3's Tableau 10 palette.
gap	Width in pixels of the white gaps between columns and between segments. Defaults to 2.
labelType	What to print inside each cell when there is room for it: "percent" (the share within the column, the default), "count" (the count or summed value), "both", or "none".
valueFormat	A d3-format specifier used to display counts or values in the labels and the tooltip (e.g. ",.0f", ".3~s"). Defaults to ",.0f".
labelFontSize	Font size of the in-cell labels. Defaults to 11.
labelCol	Optional. Color of the in-cell labels. Defaults to NULL, which picks white or near-black automatically for contrast with each cell's fill.
columnFontSize	Font size of the column (x category) labels drawn above the chart. Labels that do not fit their column are shortened or hidden. Defaults to 12.
hover	Whether to highlight the hovered cell and show a tooltip. Defaults to TRUE.
animate	Whether the columns grow into place with a short staggered animation on first render. Defaults to TRUE.
legend	Whether to display a legend of the fill categories. Defaults to TRUE.
legendTitle	Optional. Title of the legend. Defaults to NULL, which uses the name of the fill column. Use "" to hide it.
legendFontSize	Font size of the legend text. Defaults to 11.
xtitle	Optional. Title of the x-axis (bottom).
ytitle	Optional. Title of the y-axis (left).
axisTitleFontSize	Font size of the axis titles. Defaults to 12.
xFontSize	Font size of the x-axis tick labels. Defaults to 10.
yFontSize	Font size of the y-axis tick labels. Defaults to 10.
title	Optional. Title of the chart.
titleFontSize	Font size of the chart title. Defaults to 16.
font	The font family used for all text. Defaults to "Verdana, Geneva, Tahoma, sans-serif".
bgcol	Background color of the SVG. It is also the color of the gaps between cells. Defaults to "white".

axisCol	Color of the axes, the axis labels, the column labels, the legend text and the title. Defaults to "black".
width	Optional. The width of the SVG output.
height	Optional. The height of the SVG output.

Value

An SVG mosaic chart.

Examples

```
# Survival on the Titanic by passenger class (pre-aggregated counts)
titanic <- as.data.frame(Titanic)

mosaic_chart(
  data   = titanic,
  x      = "Class",
  fill   = "Survived",
  value  = "Freq",
  colors = c("#e45756", "#4c78a8"),
  xtitle = "Share of passengers and crew",
  ytitle = "Share within class",
  title  = "Who survived the Titanic?"
)

# Counting rows: gears by number of cylinders. Factor levels set the order
# of both the columns and the stacked segments.
cars <- mtcars
cars$cyl  <- factor(cars$cyl,  levels = c(8, 6, 4),
                    labels = c("8 cyl", "6 cyl", "4 cyl"))
cars$gear <- factor(cars$gear, levels = c(3, 4, 5),
                    labels = c("3 gears", "4 gears", "5 gears"))

mosaic_chart(
  data        = cars,
  x           = "cyl",
  fill        = "gear",
  labelType   = "both",
  legendTitle = "Gears",
  title       = "Motor Trend cars: gears by cylinders"
)

# Hair and eye color of statistics students (counts are summed over Sex)
hair_eye <- as.data.frame(HairEyeColor)
hair_eye$Eye <- factor(hair_eye$Eye,
                       levels = c("Brown", "Hazel", "Green", "Blue"))

mosaic_chart(
  data   = hair_eye,
  x      = "Hair",
  fill   = "Eye",
  value  = "Freq",
```

```
  colors = c("#8c564b", "#bcbd22", "#2ca02c", "#1f77b4"),
  gap    = 3,
  xtitle = "Hair color",
  title  = "Hair and eye color of 592 students"
)
```

`parliament_chart` *Display a Parliament Chart*

Description

Display a Parliament Chart

Usage

```
parliament_chart(
  data,
  categorical_column,
  numerical_column,
  seatSize = 6,
  padding = 2,
  maxRows = NULL,
  title = NULL,
  titleFontSize = 22,
  font = "Verdana, Geneva, Tahoma, sans-serif",
  bgcol = "#CAD0D3",
  width = NULL,
  height = NULL
)
```

Arguments

`data`	A data frame with a categorical column and a numerical column
`categorical_column`	The categorical column to consider
`numerical_column`	The numerical column to consider
`seatSize`	The size of each seat. Defaults to 6.
`padding`	The padding between seats. Defaults to 2.
`maxRows`	The maximum number of rows. Optional.
`title`	The title of the chart. Optional.
`titleFontSize`	Font size for the title. Defaults to 22.
`font`	Font family for text. Defaults to "Verdana, Geneva, Tahoma, sans-serif".
`bgcol`	Background color of the chart. Defaults to "#CAD0D3".
`width`	Width of the SVG canvas. Optional.
`height`	Height of the SVG canvas. Optional.

Value

A D3-rendered Parliament chart

Examples

```
political_results_example <- data.frame(
    political_party = c("SDP", "CDU", "Linke"),
    number_of_seats = c(200, 40, 30)
)
parliament_chart(
  data = political_results_example,
  categorical_column = "political_party",
  numerical_column = "number_of_seats",
  title = "German Bundestag (results not real, just an example)",
  seatSize = 10,
  bgcol = "#fefefe"
)
```

---

pie_chart *Create a pie chart*

Description

Create a pie chart

Usage

```
pie_chart(
  data,
  value,
  label,
  colorCategory = "Paired",
  innerRadius = 0,
  outerRadius = "auto",
  padRadius = 0,
  padAngle = NULL,
  cornerRadius = 0,
  labelFont = "sans-serif",
  title = NULL,
  titleFontSize = 22,
  font = "Verdana, Geneva, Tahoma, sans-serif",
  bgcol = "white",
  opacity = 1,
  labelHeight = 18,
  width = NULL,
  height = NULL
)
```

Arguments

data	The data frame to consider.
value	The numeric variable to consider.
label	The labeling variable to consider.
colorCategory	A D3 categorical color scheme, you can find more here <https://github.com/d3/d3-scale-chromatic#categorical>. Defaults to 'Paired'
innerRadius	The size of the inner radius of the pie. Defaults to 0. Set the inner radius to a higher value to plot a donut chart.
outerRadius	The size of the outer radius of the pie.
padRadius	From the D3 official documentation, The pad radius compute the fixed linear distance separating adjacent arcs, defined as padRadius * padAngle.
padAngle	Optional. From the D3 official documentation, the padAngle is used to set the padding angle between consecutive arcs.
cornerRadius	From the D3 official documentation, the value of the corner radius for rounded corners. If the corner radius is greater than zero, the corners of the arc are rounded using circles of the given radius. Defaults to 0.
labelFont	The font family of the legend. Defaults to 'sans-serif'.
title	Optional. The title of the plot.
titleFontSize	The font size of the plot title. Defaults to 22.
font	The font family to consider for the titles. Defaults to "Verdana, Geneva, Tahoma, sans-serif".
bgcol	The background color of the SVG. Defaults to "white".
opacity	The color opacity of the pie (from 0 to 1). Defaults to 1.
labelHeight	The height of the legend. Defaults to 18.
width	Optional. The width of the SVG output.
height	Optional. The height of the SVG output.

Value

A SVG pie chart

Examples

```
library(dplyr) # needed for the starwars data frame

# starwars is part of the dplyr data frame
mini_starwars <- starwars %>% tidyr::drop_na(mass) %>%
  sample_n(size = 5) # getting 5 random values

pie_chart(
  data = mini_starwars,
  value = "mass",
  label = "name"
)
```

plant_growth *Visualize Plant Growth Using D3*

Description

This function creates a plant growth meter visualization.

Usage

```
plant_growth(
  fill_level = 0.5,
  potWidth = 100,
  potHeight = 40,
  plantMaxHeight = 150,
  stemColor = "#228B22",
  potColor = "#8B4513",
  flowerColor = "#FF69B4",
  strokeColor = "#333",
  strokeWidth = 2,
  renderFillLabel = TRUE,
  titleText = "Plant Growth",
  titleColor = "#333",
  titleFontSize = "14px",
  font = "sans-serif"
)
```

Arguments

fill_level	Numeric between 0 and 1 indicating growth level.
potWidth	Width of the pot.
potHeight	Height of the pot.
plantMaxHeight	Max height of plant stem.
stemColor	Color of the plant stem and leaves.
potColor	Color of the pot.
flowerColor	Color of the flowers that bloom when growth is high.
strokeColor	Outline color for the pot.
strokeWidth	Outline width.
renderFillLabel	Whether to display a growth label.
titleText	Title shown below the pot.
titleColor	Title color.
titleFontSize	Font size of the title.
font	Font family.

pulse_grid *Create a pulse grid.*

Description

'pulse_grid()' creates a tiled grid where each cell color reflects a numeric value and the cells gently pulse according to their intensity.

Usage

```
pulse_grid(
  data,
  x,
  y,
  value,
  low = "#E0FBFC",
  high = "#0B525B",
  stroke = "white",
  strokeWidth = 1,
  cellPadding = 0.12,
  cornerRadius = 10,
  pulse = TRUE,
  pulseStrength = 0.18,
  showValues = TRUE,
  digits = 1,
  valueFontSize = 12,
  xFontSize = 12,
  yFontSize = 12,
  title = NULL,
  titleFontSize = 22,
  font = "Verdana, Geneva, Tahoma, sans-serif",
  bgcol = "#0F172A",
  labelColor = "#F8FAFC",
  titleColor = "#F8FAFC",
  width = NULL,
  height = NULL
)
```

Arguments

data	The data frame containing the grid coordinates and values.
x	The column name mapped to the grid columns.
y	The column name mapped to the grid rows.
value	The numeric column name mapped to tile intensity.
low	The low end color of the value scale. Defaults to '#E0FBFC'.

high	The high end color of the value scale. Defaults to '#0B525B'.
stroke	The border color of each tile. Defaults to 'white'.
strokeWidth	Border width of each tile. Defaults to 1.
cellPadding	Space between tiles, from 0 to 0.9. Defaults to 0.12.
cornerRadius	Corner radius of each tile. Defaults to 10.
pulse	Logical. Whether to animate the tiles. Defaults to 'TRUE'.
pulseStrength	Numeric multiplier controlling pulse amplitude. Defaults to 0.18.
showValues	Logical. Whether to print values inside tiles. Defaults to 'TRUE'.
digits	Number of digits used for in-cell labels. Defaults to 1.
valueFontSize	Font size of in-cell labels. Defaults to 12.
xFontSize	Font size of the column labels. Defaults to 12.
yFontSize	Font size of the row labels. Defaults to 12.
title	Optional. The title of the plot.
titleFontSize	Font size of the plot title. Defaults to 22.
font	The font family used in labels and title. Defaults to 'Verdana, Geneva, Tahoma, sans-serif'.
bgcol	Background color of the SVG. Defaults to '#0F172A'.
labelColor	Color of the row and column labels. Defaults to '#F8FAFC'.
titleColor	Color of the title. Defaults to '#F8FAFC'.
width	Optional. The width of the SVG output.
height	Optional. The height of the SVG output.

Value

A SVG pulse grid.

Examples

```
pulse_data <- expand.grid(
  hour = c("09:00", "12:00", "15:00", "18:00"),
  team = c("North", "South", "East")
)
pulse_data$load <- c(0.35, 0.72, 0.56, 0.91,
                     0.41, 0.67, 0.83, 0.58,
                     0.28, 0.49, 0.77, 0.63)

pulse_grid(
  data = pulse_data,
  x = "hour",
  y = "team",
  value = "load",
  title = "Team Activity Pulse Grid",
  low = "#D9F0FF",
  high = "#EE6C4D",
  bgcol = "#0B132B"
)
```

`radar_chart` *Create a radar chart.*

Description

A radar chart (also called a spider or web chart) compares several quantitative variables at once by placing each one on its own spoke radiating from a common center. Each series (a row of the data) is drawn as a closed, translucent polygon joining its values on every spoke, so series can be compared by their overall shape and footprint. Concentric grid rings mark the scale. The look and feel is inspired by the radar chart of the Chart.js JavaScript library, reimplemented here in plain D3.

The chart is interactive: the polygons grow out from the center on first render, hovering a point shows a tooltip with its value while the other series fade, hovering a polygon focuses that series, and clicking a legend entry toggles the corresponding series on or off.

Usage

```
radar_chart(
  data,
  group,
  axes = NULL,
  axisLabels = NULL,
  maxValue = NULL,
  scaleEachAxis = FALSE,
  levels = 5,
  gridShape = c("polygon", "circle"),
  colors = NULL,
  fillOpacity = 0.2,
  lineWidth = 2,
  pointRadius = 3,
  showTicks = TRUE,
  valueFormat = ",",
  gridCol = "#dddddd",
  hover = TRUE,
  animate = TRUE,
  legend = TRUE,
  legendFontSize = 12,
  labelFontSize = 12,
  tickFontSize = 10,
  title = NULL,
  titleFontSize = 16,
  font = "Verdana, Geneva, Tahoma, sans-serif",
  bgcol = "white",
  axisCol = "black",
  width = NULL,
  height = NULL
)
```

Arguments

data	The data frame containing the variables to consider, in wide format (one row per series).
group	The name of the column holding the series labels. Its values must be unique (one row per series).
axes	Optional. A character vector of at least three numeric column names, each drawn as a spoke of the radar. Defaults to NULL, which uses every numeric column except group.
axisLabels	Optional. A character vector, the same length as axes, of labels displayed at the end of each spoke. Defaults to the column names.
maxValue	Optional. The value of the outer ring. With a shared scale, a single positive number; with scaleEachAxis = TRUE, either a single number or one number per axis. Defaults to NULL, which rounds the data maximum up to a "nice" value. Values above maxValue are drawn on the outer ring.
scaleEachAxis	Whether every axis gets its own scale running from 0 to that axis' (nice) maximum. Useful when the axes are measured in different units. The ring tick labels are then hidden and each axis label shows its maximum instead. Defaults to FALSE (one scale shared by all axes).
levels	The number of concentric grid rings. Defaults to 5.
gridShape	Shape of the grid rings, either "polygon" (default) or "circle".
colors	Optional. A vector of colors, one per series (recycled if shorter). Defaults to NULL, which uses a soft palette in the spirit of Chart.js.
fillOpacity	Opacity of the polygon fills, between 0 and 1. Defaults to 0.2.
lineWidth	Width of the polygon outlines in pixels. Defaults to 2.
pointRadius	Radius of the point markers in pixels. Defaults to 3. Use 0 to hide the markers.
showTicks	Whether to print the value of each grid ring along the top spoke (shared scale only). Defaults to TRUE.
valueFormat	A d3-format specifier used to display values in the ticks and the tooltip, e.g. ".1f" or ".0%". Defaults to "," (thousands separator, insignificant zeros trimmed).
gridCol	Color of the grid rings and spokes. Defaults to "#dddddd".
hover	Whether to show a tooltip and highlight the hovered series. Defaults to TRUE.
animate	Whether the polygons grow out from the center on first render. Defaults to TRUE.
legend	Whether to display a clickable legend of the series. Defaults to TRUE.
legendFontSize	Font size of the legend text. Defaults to 12.
labelFontSize	Font size of the axis labels. Defaults to 12.
tickFontSize	Font size of the grid ring tick labels. Defaults to 10.
title	Optional. Title of the chart.
titleFontSize	Font size of the chart title. Defaults to 16.
font	The font family used for all text. Defaults to "Verdana, Geneva, Tahoma, sans-serif".

bgcol	Background color of the SVG. Defaults to "white".
axisCol	Color of the axis labels, tick labels, legend text and title. Defaults to "black".
width	Optional. The width of the SVG output.
height	Optional. The height of the SVG output.

Value

An SVG radar chart.

Examples

```
# Skill ratings of three football players (shared 0-100 scale)
players <- data.frame(
  player    = c("Ana", "Ben", "Chloe"),
  speed     = c(88, 72, 65),
  power     = c(60, 91, 70),
  passing   = c(79, 58, 93),
  dribbling = c(92, 64, 81),
  defending = c(45, 83, 74),
  stamina   = c(70, 77, 88)
)

radar_chart(
  data     = players,
  group    = "player",
  axes     = c("speed", "power", "passing", "dribbling",
               "defending", "stamina"),
  maxValue = 100,
  title    = "Player skill ratings"
)

# Variables measured in different units: give every axis its own scale
cyl_means <- aggregate(cbind(mpg, hp, wt, qsec, disp) ~ cyl, data = mtcars,
                  FUN = mean)
cyl_means$cyl <- paste(cyl_means$cyl, "cylinders")

radar_chart(
  data          = cyl_means,
  group         = "cyl",
  axisLabels    = c("Miles/gallon", "Horsepower", "Weight (1000 lbs)",
                    "1/4 mile time (s)", "Displacement (cu.in.)"),
  scaleEachAxis = TRUE,
  levels        = 4,
  gridShape     = "circle",
  valueFormat   = ".1f",
  title         = "mtcars averages by number of cylinders"
)
```

`radial_bar_chart` *Create a radial bar chart.*

Description

A radial bar chart (sometimes called a multi-ring progress chart or "activity rings") draws each category as a concentric circular track. A faint background track shows the full scale, and a colored arc with rounded ends sweeps along it in proportion to the category's value relative to `max`. It is a compact way to compare a handful of progress or completion rates. The look is inspired by the radialBar chart of the ApexCharts JavaScript library, reimplemented here in plain D3.

The first row of `data` is drawn as the outermost ring. The arcs sweep in on load, each ring is labelled at the start of its track, and the center shows the average (or total) of the values. Hovering a ring highlights it, replaces the center text with that ring's name and value, and shows a tooltip.

By default the rings cover 270 degrees (`startAngle = 0`, `endAngle = 270`), leaving the top-left quadrant free for the ring labels. Angles are in degrees, measured clockwise from 12 o'clock. Use `endAngle = 360` for full circles (labels then follow the curve of each track), or for example `startAngle = -90, endAngle = 90` for a semicircle gauge.

Usage

```
radial_bar_chart(
  data,
  label,
  value,
  max = 100,
  startAngle = 0,
  endAngle = 270,
  colors = NULL,
  trackCol = "#ececec",
  hollow = 0.35,
  ringGap = 0.3,
  roundedCaps = TRUE,
  showLabels = TRUE,
  showValues = TRUE,
  labelFontSize = 12,
  centerValue = c("average", "total", "none"),
  centerLabel = NULL,
  centerFontSize = NULL,
  valueFormat = ",.0f",
  suffix = NULL,
  hover = TRUE,
  animate = TRUE,
  duration = 1200,
  title = NULL,
  titleFontSize = 16,
  font = "Verdana, Geneva, Tahoma, sans-serif",
```

```
  bgcol = "white",
  textCol = "#333333",
  width = NULL,
  height = NULL
)
```

Arguments

data	The data frame containing the variables to consider (one row per ring).
label	The name of the column holding the ring labels.
value	The name of the numeric (non-negative) column holding the ring values.
max	The value that corresponds to a full sweep of the track. Values above max are capped at a full sweep (with a warning). Defaults to 100.
startAngle	Angle in degrees at which every track starts, measured clockwise from 12 o'clock. Defaults to 0.
endAngle	Angle in degrees at which every track ends. Must be greater than startAngle and at most startAngle + 360. Defaults to 270.
colors	Optional. A vector of colors, one per ring (recycled if shorter). Defaults to NULL, which uses a built-in palette.
trackCol	Color of the background tracks. Defaults to "#ececec".
hollow	Size of the empty center as a fraction (0 to 0.9) of the outer radius. Defaults to 0.35.
ringGap	Fraction (0 to 0.9) of each ring's band left empty between neighbouring tracks. Defaults to 0.3.
roundedCaps	Whether the arcs (and partial tracks) have rounded ends. Defaults to TRUE.
showLabels	Whether to label each ring at the start of its track. Defaults to TRUE.
showValues	Whether to append the formatted value to each ring label (e.g. "Vimeo: 76%"). Defaults to TRUE.
labelFontSize	Font size of the ring labels. Defaults to 12.
centerValue	What to show in the center when nothing is hovered: "average" (default), "total" or "none".
centerLabel	Optional. Text shown above the center value. Defaults to "Average" or "Total" depending on centerValue.
centerFontSize	Optional. Font size of the big center value. Defaults to NULL, which sizes it to fit the hollow.
valueFormat	A d3-format specifier used to display values. Defaults to ",.0f".
suffix	Optional. Text appended to every displayed value. Defaults to NULL, which uses "%" when max = 100 and nothing otherwise.
hover	Whether hovering a ring highlights it, updates the center text and shows a tooltip. Defaults to TRUE.
animate	Whether the arcs sweep in on first render. Defaults to TRUE.
duration	Duration of the sweep animation in milliseconds. Defaults to 1200.

title	Optional. Title of the chart.
titleFontSize	Font size of the chart title. Defaults to 16.
font	The font family used for all text. Defaults to "Verdana, Geneva, Tahoma, sans-serif".
bgcol	Background color of the SVG. Defaults to "white".
textCol	Color of the title, labels and center text. Defaults to "#333333".
width	Optional. The width of the SVG output.
height	Optional. The height of the SVG output.

Value

An SVG radial bar chart.

Examples

```
# Share of users active on each platform (percentages)
platforms <- data.frame(
  platform = c("Vimeo", "Messenger", "Facebook", "LinkedIn"),
  share    = c(76, 67, 61, 90)
)

radial_bar_chart(
  data  = platforms,
  label = "platform",
  value = "share",
  title = "Weekly active users (share of total)"
)

# Full circles, total in the center, custom colors and a custom maximum
sales <- data.frame(
  region = c("North", "South", "East", "West", "Central"),
  units  = c(420, 310, 275, 190, 120)
)

radial_bar_chart(
  data        = sales,
  label       = "region",
  value       = "units",
  max         = 500,
  endAngle    = 360,
  centerValue = "total",
  centerLabel = "Units sold",
  colors      = c("#1b9e77", "#d95f02", "#7570b3", "#e7298a", "#66a61e"),
  title       = "Units sold by region (target: 500)"
)

# Gauge layout open at the bottom, without values in the ring labels
radial_bar_chart(
  data       = platforms,
  label      = "platform",
```

```
  value      = "share",
  startAngle = -135,
  endAngle   = 135,
  hollow     = 0.45,
  showValues = FALSE,
  trackCol   = "#f2f2f2"
)
```

`ridgeline_plot` *Create a ridgeline plot (joy plot).*

Description

A ridgeline plot (also known as a "joy plot") shows the distribution of a numeric variable for several groups as a stack of partially overlapping density curves that share a common x-axis. The nickname comes from the cover of Joy Division's 1979 album *Unknown Pleasures*, itself a reproduction of a 1970 plot of successive radio pulses from the first discovered pulsar. The format was popularised by Claus Wilke's `ggridges` package and by FiveThirtyEight and The Washington Post graphics desks.

Ridgeline plots are ideal for comparing the shape of many distributions at once (e.g. across months, categories, or experimental conditions) while using far less vertical space than a set of separate histograms.

The kernel density estimates are computed in the browser. Ridges are drawn from the back (top) to the front (bottom), so lower ridges overlap the ones above them. Hovering a ridge (or its label) brings it to the front, dims the others, and shows a small label with the group's number of observations and mean.

Usage

```
ridgeline_plot(
  data,
  x,
  group,
  bandwidth = NULL,
  kernel = c("gaussian", "epanechnikov"),
  overlap = 1.5,
  fill = "steelblue",
  fillOpacity = 0.8,
  lineCol = "#333333",
  lineWidth = 1,
  sort = c("none", "mean", "median"),
  stat = c("none", "mean", "median"),
  statColor = "#d62728",
  hover = TRUE,
  xtitle = NULL,
  xtitleFontSize = 13,
  title = NULL,
  titleFontSize = 16,
```

```
  xFontSize = 11,
  yFontSize = 11,
  font = "Verdana, Geneva, Tahoma, sans-serif",
  bgcol = "white",
  axisCol = "black",
  width = NULL,
  height = NULL
)
```

Arguments

data	The data frame containing the variables to consider.
x	The name of the numeric column whose distribution is displayed.
group	The name of the categorical column used to split the data into one ridge per group. When the column is a factor, its level order is used (unless sort is set).
bandwidth	Optional. The kernel bandwidth, expressed as the standard deviation of the kernel (as in stats::density()). When NULL (default), a single joint bandwidth is computed from the pooled data with Silverman's rule of thumb (equivalent to stats::bw.nrd0()), so that all ridges are smoothed identically and remain comparable.
kernel	The smoothing kernel, either "gaussian" (default) or "epanechnikov".
overlap	How much each ridge may overlap the one above it, in units of the vertical distance between two consecutive baselines. The tallest peak of the chart reaches exactly overlap baselines up. Defaults to 1.5; use 1 for ridges that just touch and larger values for a more dramatic, Joy Division-like look.
fill	Fill color of the ridges. Either a single color (default "steelblue") applied to every ridge, an unnamed vector of colors recycled over the groups (in their original order), or a named vector whose names match the group values.
fillOpacity	Fill opacity of the ridges (0 to 1). Defaults to 0.8.
lineCol	Color of the density outline. Defaults to "#333333".
lineWidth	Width of the density outline in pixels. Defaults to 1.
sort	How to order the ridges from top to bottom. One of "none" (default, factor level order or order of appearance), "mean", or "median". When sorting, the group with the smallest mean/median is placed at the top.
stat	Character. Draws a vertical line inside each ridge at the "mean" or "median" of its values. Defaults to "none" (no line).
statColor	Color of the mean/median line. Defaults to "#d62728".
hover	Logical. When TRUE (default), hovering a ridge or its label raises and highlights it and shows its number of observations and mean.
xtitle	Optional. Title of the x-axis.
xtitleFontSize	Font size of the x-axis title. Defaults to 13.
title	Optional. Title of the chart.
titleFontSize	Font size of the chart title. Defaults to 16.

xFontSize	Font size of the x-axis tick labels. Defaults to 11.
yFontSize	Font size of the group labels. Defaults to 11.
font	The font family used for all text. Defaults to "Verdana, Geneva, Tahoma, sans-serif".
bgcol	Background color of the SVG. Defaults to "white".
axisCol	Color of axis lines, ticks, and labels. Defaults to "black".
width	Optional. The width of the SVG output.
height	Optional. The height of the SVG output.

Value

An SVG ridgeline plot.

Examples

```
# Simulated reaction times for five experimental conditions
set.seed(2026)
reaction <- data.frame(
  time = c(rnorm(200, 300, 40), rnorm(200, 340, 50),
           rnorm(200, 380, 35), c(rnorm(100, 360, 25), rnorm(100, 460, 30)),
           rnorm(200, 420, 60)),
  condition = rep(c("Baseline", "Noise", "Fatigue", "Dual task", "Night"),
                  each = 200)
)

ridgeline_plot(
  data   = reaction,
  x      = "time",
  group  = "condition",
  sort   = "mean",
  stat   = "median",
  title  = "Reaction time by condition",
  xtitle = "Reaction time (ms)"
)

# Highway fuel economy by vehicle class
library(ggplot2) # for the mpg dataset

ridgeline_plot(
  data        = mpg,
  x           = "hwy",
  group       = "class",
  sort        = "median",
  fill        = "#2c7fb8",
  lineCol     = "white",
  fillOpacity = 0.9,
  overlap     = 2,
  title       = "Highway fuel economy by vehicle class",
  xtitle      = "Highway miles per gallon"
)
```

```
# One color per group with a named vector, Epanechnikov kernel
ridgeline_plot(
  data    = iris,
  x       = "Petal.Length",
  group   = "Species",
  kernel  = "epanechnikov",
  fill    = c(setosa = "#66c2a5", versicolor = "#fc8d62",
              virginica = "#8da0cb"),
  overlap = 1,
  stat    = "mean",
  xtitle  = "Petal length (cm)"
)
```

`rose_chart` *Create a Nightingale rose chart.*

Description

A Nightingale rose (also called a coxcomb or polar area chart) divides a circle into equal-angle wedges, one per category, and encodes each value by how far its wedge extends from the center. The form is credited to Florence Nightingale, who used it in 1858 to show that far more British soldiers in the Crimean War died of preventable disease than of wounds.

Radii follow a square-root scale so that the *area* of each wedge, not its length, is proportional to the value, keeping the visual honest. When a `group` column is supplied, each wedge is split into stacked segments whose areas are again proportional to their values. Dashed radial grid circles carry value labels, and category labels sit outside the wedges. The look is inspired by the polar-interval "rose" chart of the AntV G2 library, reimplemented in plain D3.

The chart is interactive: wedges grow out from the center on first render, hovering a wedge pops it outward and shows a tooltip (with a per-group breakdown when stacked), and, when stacked, hovering a legend entry focuses that group while clicking it toggles the group on or off.

Usage

```
rose_chart(
  data,
  category,
  value,
  group = NULL,
  colors = NULL,
  sort = c("none", "desc", "asc"),
  innerRadius = 0,
  startAngle = 0,
  padAngle = 0,
  cornerRadius = 0,
  opacity = 0.9,
  strokeCol = "white",
```

```
  strokeWidth = 1,
  gridLines = 4,
  gridCol = "#e0e0e0",
  gridFontSize = 10,
  showLabels = TRUE,
  labelFontSize = 11,
  valueFormat = ",.0f",
  hover = TRUE,
  animate = TRUE,
  legend = TRUE,
  legendFontSize = 11,
  title = NULL,
  titleFontSize = 16,
  font = "Verdana, Geneva, Tahoma, sans-serif",
  bgcol = "white",
  axisCol = "black",
  width = NULL,
  height = NULL
)
```

Arguments

data	The data frame containing the variables to consider. Rows that share the same category (and group) are summed.
category	The name of the column holding the categories (one wedge per category). If it is a factor, its levels set the order of the wedges; otherwise they appear in the order found in the data.
value	The name of the numeric column holding the (non-negative) values.
group	Optional. The name of a categorical column used to split each wedge into stacked segments. If it is a factor, its levels set the stacking order (first level innermost). Defaults to NULL.
colors	Optional. A vector of colors, one per group when group is supplied, otherwise one per category (recycled if needed). A single color paints every wedge alike. Defaults to NULL, which uses an AntV-style ten-color palette.
sort	How to order the wedges: "none" (data or factor order, the default), "desc" or "asc" by category total.
innerRadius	Size of an empty hole in the middle, as a fraction of the outer radius (between 0 and 0.9). Defaults to 0.
startAngle	Angle, in degrees clockwise from 12 o'clock, at which the first wedge starts. Defaults to 0.
padAngle	Gap between adjacent wedges, in radians. Defaults to 0.
cornerRadius	Corner rounding of the wedges, in pixels. Defaults to 0.
opacity	Fill opacity of the wedges (0 to 1). Defaults to 0.9.
strokeCol	Color of the wedge outlines. Defaults to "white".
strokeWidth	Width of the wedge outlines in pixels. Defaults to 1.

gridLines	Approximate number of radial grid circles (value rings). Use 0 to hide them. Defaults to 4.
gridCol	Color of the grid circles and spokes. Defaults to "#e0e0e0".
gridFontSize	Font size of the grid value labels. Defaults to 10.
showLabels	Whether to display the category labels around the chart. Defaults to TRUE.
labelFontSize	Font size of the category labels. Defaults to 11.
valueFormat	A d3-format specifier used to display values in the tooltip (e.g. ",.0f", ",.1f", "$,.2s"). Defaults to ",.0f".
hover	Whether hovering a wedge pops it outward and shows a tooltip. Defaults to TRUE.
animate	Whether wedges grow out from the center on first render. Defaults to TRUE.
legend	Whether to display a legend of the groups (only used when group is supplied). Defaults to TRUE.
legendFontSize	Font size of the legend text. Defaults to 11.
title	Optional. Title of the chart.
titleFontSize	Font size of the chart title. Defaults to 16.
font	The font family used for all text. Defaults to "Verdana, Geneva, Tahoma, sans-serif".
bgcol	Background color of the SVG. Defaults to "white".
axisCol	Color of the title, labels and grid value labels. Defaults to "black".
width	Optional. The width of the SVG output.
height	Optional. The height of the SVG output.

Value

An SVG Nightingale rose chart.

Examples

```
# Florence Nightingale's data: deaths in the British army in the East,
# April 1854 to March 1855, by cause (as in the HistData package)
mon <- c("Apr 1854", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov",
           "Dec", "Jan 1855", "Feb", "Mar")
crimea <- data.frame(
  month = factor(rep(mon, times = 3), levels = mon),
  cause = factor(rep(c("Preventable disease", "Wounds", "Other causes"),
                     each = 12),
                 levels = c("Preventable disease", "Wounds", "Other causes")),
  deaths = c(1, 12, 11, 359, 828, 788, 503, 844, 1725, 2761, 2120, 1205,
             0, 0, 0, 0, 1, 81, 132, 287, 114, 83, 42, 32,
             5, 9, 6, 23, 30, 70, 128, 106, 131, 324, 361, 172)
)

rose_chart(
  data     = crimea,
```

```
  category = "month",
  value    = "deaths",
  group    = "cause",
  colors   = c("#7fa7c9", "#e0645c", "#4d4d4d"),
  title    = "Causes of death in the army in the East, 1854-55",
  height   = 560
)

# A simple rose: one wedge per category, sorted by value
rose_chart(
  data     = data.frame(state = rownames(USArrests)[1:12],
                        murder = USArrests$Murder[1:12]),
  category = "state",
  value    = "murder",
  sort     = "desc",
  valueFormat = ".1f",
  title    = "Murder arrests per 100,000 residents (1973)"
)

# Donut-style rose with a hole, gaps and rounded corners
rose_chart(
  data         = as.data.frame(table(cyl = mtcars$cyl, gear = mtcars$gear)),
  category     = "gear",
  value        = "Freq",
  group        = "cyl",
  innerRadius  = 0.25,
  padAngle     = 0.03,
  cornerRadius = 4,
  title        = "Cars by number of gears and cylinders"
)
```

`scatter_plot` *Create a scatter plot.*

Description

Create a scatter plot.

Usage

```
scatter_plot(
  data,
  x,
  y,
  col = "crimson",
  size = 2,
  xticks = NULL,
  yticks = NULL,
  xtitle = NULL,
```

```
  xtitleFontSize = 16,
  ytitle = NULL,
  ytitleFontSize = 16,
  title = NULL,
  titleFontSize = 22,
  stroke = NULL,
  strokeWidth = NULL,
  font = "Verdana, Geneva, Tahoma, sans-serif",
  bgcol = "#CAD0D3",
  opacity = 1,
  axisCol = "black",
  width = NULL,
  height = NULL
)
```

Arguments

data	The data frame containing the quantitative variables.
x	The x-variable to consider.
y	The y-variable to consider.
col	The color of the dots. Defaults to 'crimson'.
size	The size of the dots. Defaults to 2.
xticks	Optional. The number of x-axis ticks to consider.
yticks	Optional. The number of y-axis ticks to consider.
xtitle	Optional. the title of the x-axis.
xtitleFontSize	The font size of the x-axis title. Defaults to 16.
ytitle	Optional. The title of the y-axis.
ytitleFontSize	The font size of the y-axis title. Defaults to 16.
title	Optional. the title of the plot.
titleFontSize	The font size of the plot title. Defaults to 22.
stroke	Optional. the stroke color of the dots.
strokeWidth	Optional. the stroke width of the dots.
font	The font family to consider for the titles. Defaults to "Verdana, Geneva, Tahoma, sans-serif".
bgcol	The background color of the SVG. Defaults to "#CAD0D3" HEX color.
opacity	The color opacity of the dots (from 0 to 1). Defaults to 1.
axisCol	the color of the x and y axis. It includes the ticks, the labels and titles. Defaults to 'black'.
width	Optional. the width of the SVG output.
height	Optional. the height of the SVG output.

Value

A SVG scatter plot.

Examples

```
scatter_plot(
  data = mtcars,
  x = "mpg",
  y = "wt"
  )
```

`slope_chart` *Create a slope chart.*

Description

A slope chart (or slopegraph) compares two values per category by placing them on two vertical axes, typically "before" and "after" or two years, and connecting them with a straight line. The steepness and direction of each line show at a glance which categories rose, which fell, and how their ranking changed. The form was popularised by Edward Tufte in *The Visual Display of Quantitative Information* (1983) and is a staple of the New York Times graphics desk.

Each line is labelled with its category name and value at both ends; labels that would overlap are nudged apart vertically. Lines can be colored by direction (increase, decrease, no change), a subset of categories can be emphasised with `highlight`, and hovering over a line fades the others.

Usage

```
slope_chart(
  data,
  start,
  end,
  group,
  startLabel = NULL,
  endLabel = NULL,
  colorByDirection = TRUE,
  upCol = "#1a9850",
  downCol = "#d73027",
  flatCol = "#888888",
  lineCol = "steelblue",
  highlight = NULL,
  mutedCol = "#cccccc",
  lineWidth = 2,
  circleRadius = 4,
  showValues = TRUE,
  digits = NULL,
  labelFontSize = 11,
  headerFontSize = 13,
  title = NULL,
  titleFontSize = 16,
  font = "Verdana, Geneva, Tahoma, sans-serif",
```

```
  bgcol = "white",
  axisCol = "black",
  width = NULL,
  height = NULL
)
```

Arguments

data	The data frame containing the variables to consider, in wide format (one row per category).
start	The name of the numeric column holding the values plotted on the left axis.
end	The name of the numeric column holding the values plotted on the right axis.
group	The name of the column holding the category labels (one line per category).
startLabel	Header displayed above the left axis. Defaults to the value of start.
endLabel	Header displayed above the right axis. Defaults to the value of end.
colorByDirection	Whether to color lines by direction of change (upCol, downCol, flatCol). If FALSE, every line uses lineCol. Defaults to TRUE.
upCol	Color of lines that increase. Defaults to "#1a9850".
downCol	Color of lines that decrease. Defaults to "#d73027".
flatCol	Color of lines with no change. Defaults to "#888888".
lineCol	Color of all lines when colorByDirection = FALSE. Defaults to "steelblue".
highlight	Optional. A character vector of categories (values of the group column) to emphasise. All other lines are drawn in mutedCol. Defaults to NULL (no emphasis).
mutedCol	Color of non-highlighted lines when highlight is used (their labels are drawn semi-transparent). Defaults to "#cccccc".
lineWidth	Width of the lines in pixels. Defaults to 2.
circleRadius	Radius of the end-point dots in pixels. Defaults to 4. Use 0 to hide the dots.
showValues	Whether to print the value next to each category label. Defaults to TRUE.
digits	Optional. Number of decimal places used to print the values. Defaults to NULL, which prints the values as they are.
labelFontSize	Font size of the category and value labels. Defaults to 11.
headerFontSize	Font size of the two axis headers. Defaults to 13.
title	Optional. Title of the chart.
titleFontSize	Font size of the chart title. Defaults to 16.
font	The font family used for all text. Defaults to "Verdana, Geneva, Tahoma, sans-serif".
bgcol	Background color of the SVG. Defaults to "white".
axisCol	Color of the two vertical axes, the headers, the title and the labels. Defaults to "black".
width	Optional. The width of the SVG output.
height	Optional. The height of the SVG output.

Value

An SVG slope chart.

Examples

```
# Current receipts of government as a percentage of GDP, 1970 vs 1979
# (adapted from Edward Tufte's classic slopegraph)
receipts <- data.frame(
  country = c("Sweden", "Netherlands", "Norway", "Britain", "France",
              "Germany", "Belgium", "Canada", "Finland", "Italy",
              "United States", "Greece", "Switzerland", "Spain", "Japan"),
  y1970 = c(46.9, 44.0, 43.5, 40.7, 39.0, 37.5, 35.2, 35.2, 34.9, 30.4,
            30.3, 26.8, 26.5, 22.5, 20.7),
  y1979 = c(57.4, 55.8, 52.2, 39.0, 43.4, 42.9, 43.2, 35.8, 38.2, 35.7,
            32.5, 30.6, 33.2, 27.1, 26.6)
)

slope_chart(
  data       = receipts,
  start      = "y1970",
  end        = "y1979",
  group      = "country",
  startLabel = "1970",
  endLabel   = "1979",
  digits     = 1,
  title      = "Government receipts as % of GDP",
  height     = 520
)

# Life expectancy in 1952 vs 2007, highlighting two countries
library(gapminder)
library(dplyr)
library(tidyr)

countries <- c("China", "India", "Japan", "Brazil", "Nigeria",
               "Egypt", "Germany", "United States", "South Africa")

life <- filter(gapminder, year %in% c(1952, 2007), country %in% countries)
life <- select(life, country, year, lifeExp)
life <- pivot_wider(life, names_from = year, values_from = lifeExp,
                    names_prefix = "y")

slope_chart(
  data       = life,
  start      = "y1952",
  end        = "y2007",
  group      = "country",
  startLabel = "1952",
  endLabel   = "2007",
  highlight  = c("China", "South Africa"),
  digits     = 1,
  title      = "Life expectancy: 1952 vs 2007"
```

```
)
```

`stacked_area_chart` *Create a stacked area chart*

Description

Create a stacked area chart

Usage

```
stacked_area_chart(
  data,
  x,
  colorCategory = "Category10",
  curve = "curveLinear",
  stroke = NULL,
  strokeWidth = 1.5,
  xticks = NULL,
  yticks = NULL,
  xtitle = NULL,
  xtitleFontSize = 16,
  ytitle = NULL,
  ytitleFontSize = 16,
  title = NULL,
  titleFontSize = 22,
  font = "Verdana, Geneva, Tahoma, sans-serif",
  bgcol = "#CAD0D3",
  opacity = 1,
  axisCol = "black",
  legendBoxSize = 18,
  legendTextSize = 18,
  width = NULL,
  height = NULL
)
```

Arguments

`data`	The data frame containing the variables to consider.
`x`	The x-variable to consider. Must be a date variable in 'yyyy-mm-dd' format.
`colorCategory`	A D3 categorical color scheme, you can find more here <https://github.com/d3/d3-scale-chromatic#categorical>. Defaults to 'Category10'.
`curve`	The line's curve type to render. A complete list can be found here <https://github.com/d3/d3-shape#curves>. Defaults to 'curveLinear'.
`stroke`	Optional. The color of the stroke of the area.
`strokeWidth`	The width of the line. Defaults to 1.5.

xticks	Optional. the number of x-axis ticks to consider.
yticks	Optional. The number of y-axis ticks to consider.
xtitle	Optional. The title of the x-axis.
xtitleFontSize	The font size of the x-axis title. Defaults to 16.
ytitle	Optional. The title of the y-axis.
ytitleFontSize	The font size of the y-axis title. Defaults to 16.
title	Optional. The title of the plot.
titleFontSize	The font size of the plot title. Defaults to 22.
font	The font family to consider for the titles. Defaults to "Verdana, Geneva, Tahoma, sans-serif".
bgcol	The background color of the SVG. Defaults to "#CAD0D3" HEX color.
opacity	The color opacity of the area chart (from 0 to 1). Defaults to 1.
axisCol	the color of the x and y axis. It includes the ticks, the labels and titles. Defaults to 'black'.
legendBoxSize	The size of the legend rectangles. Defaults to 18.
legendTextSize	The font size of the legend text Defaults to 18.
width	Optional. The width of the SVG output.
height	Optional. The height of the SVG output.

Value

a SVG stacked area chart

Examples

```
data <- data.frame(
  date = c(
    "2000-01-01", "2000-02-01", "2000-03-01", "2000-04-01",
    "2000-05-01", "2000-06-01", "2000-07-01",
    "2000-08-01", "2000-09-01", "2000-10-01"
  ),
  Trade = c(
    2000,1023, 983, 2793, 1821, 1837, 1792, 1853, 791, 739
  ),
  Manufacturing = c(
    734, 694, 739, 736, 685, 621, 708, 685, 667, 693
  ),
  Leisure = c(
    1782, 1779, 1789, 658, 675, 833, 786, 675, 636, 691
  ),
  Agriculture = c(
    655, 587,623, 517, 561, 2545, 636, 584, 559, 2504
  )
)

stacked_area_chart(
```

```
  data = data,
  x = "date",
  legendTextSize = 14,
  curve = "curveCardinal",
  colorCategory = "Accent",
  bgcol = "white",
  stroke = "black",
  strokeWidth = 1
)
```

`streamgraph` *Create a streamgraph.*

Description

A streamgraph is a stacked area chart whose baseline is not fixed at zero but displaced around a central axis, so the layers flow like a river. It shows how the composition of a total changes over time while keeping attention on the shape of each layer rather than on exact values. The form was popularised by Lee Byron and Martin Wattenberg's "Stacked Graphs: Geometry & Aesthetics" (2008); this implementation reproduces the look and feel of the Highcharts streamgraph in plain D3.

Unlike `stacked_area_chart()`, which takes wide data and stacks areas on a zero baseline, `streamgraph()` takes long (tidy) data, uses a centred baseline (`offset`), orders layers inside out so the largest layers sit in the middle, and labels each layer inline at its thickest point instead of relying on a separate legend.

The chart is interactive: hovering a layer highlights it while the others dim, a vertical crosshair snaps to the nearest x value and shows a tooltip with the value of every layer at that x, and (when `legend = TRUE`) hovering a legend item focuses its layer while clicking it hides or shows the layer, re-stacking the stream with an animation. Missing x/group combinations are treated as zero.

Usage

```
streamgraph(
  data,
  x,
  group,
  value,
  offset = c("wiggle", "silhouette", "expand"),
  order = c("insideOut", "none", "reverse", "ascending", "descending"),
  curve = c("basis", "catmullRom", "monotoneX", "linear", "step"),
  colors = NULL,
  colorPalette = "Tableau10",
  opacity = 0.9,
  strokeCol = "white",
  strokeWidth = 0.5,
  labels = TRUE,
```

```
  labelFontSize = 11,
  labelCol = NULL,
  tooltip = TRUE,
  hover = TRUE,
  animate = TRUE,
  legend = TRUE,
  legendFontSize = 11,
  valueFormat = ",.2~f",
  xFormat = NULL,
  xticks = NULL,
  xFontSize = 11,
  gridCol = "#e6e6e6",
  xtitle = NULL,
  xtitleFontSize = 13,
  title = NULL,
  titleFontSize = 16,
  font = "Verdana, Geneva, Tahoma, sans-serif",
  bgcol = "white",
  axisCol = "black",
  width = NULL,
  height = NULL
)
```

Arguments

`data`	The data frame containing the variables to consider, in long format (one row per x value and group).
`x`	The name of the column holding the x values. Must be numeric (e.g. years), `Date` or `POSIXct`.
`group`	The name of the column holding the layer (series) names. If it is a factor, its level order is used as the layer order when `order = "none"` and in the legend.
`value`	The name of the numeric, non-negative column holding the thickness of each layer. Rows sharing the same `x` and `group` are summed. `NA` values are treated as zero.
`offset`	The baseline of the stream. `"wiggle"` (default) minimises the weighted change in slope of the layers (the classic streamgraph), `"silhouette"` centres the stream symmetrically around zero, and `"expand"` normalises every x to 100% (a percent axis is then shown on the left).
`order`	The order in which layers are stacked. `"insideOut"` (default) puts the layers that peak earliest in the middle and the later ones on the outside, which works best with `"wiggle"`. `"none"` keeps the order of the `group` column (factor levels or first appearance), `"reverse"` reverses it, and `"ascending"` / `"descending"` sort the layers by their total.
`curve`	The curve used to interpolate the layers: `"basis"` (default, smooth B-spline that never lets layers cross), `"catmullRom"`, `"monotoneX"`, `"linear"` or `"step"`.
`colors`	Optional. A vector of colors, one per layer (recycled if shorter). Defaults to `NULL`, which uses `colorPalette`.

colorPalette	D3 categorical color scheme used when colors is NULL (e.g. "Tableau10", "Category10", "Set2", "Paired"). Defaults to "Tableau10".
opacity	Fill opacity of the layers (0 to 1). Defaults to 0.9.
strokeCol	Color of the thin line separating the layers. Defaults to "white".
strokeWidth	Width of the separating line in pixels. Defaults to 0.5.
labels	Whether to write each layer's name inside the layer, at its thickest point. Labels that do not fit their layer are hidden. Defaults to TRUE.
labelFontSize	Font size of the inline layer labels. Defaults to 11.
labelCol	Optional. Color of the inline labels. Defaults to NULL, which picks white or near-black automatically for contrast with each layer's fill.
tooltip	Whether to show the vertical crosshair and the tooltip listing every layer's value at the hovered x. Defaults to TRUE.
hover	Whether hovering a layer (or a legend item) highlights it and dims the others. Defaults to TRUE.
animate	Whether the stream is revealed from left to right on first render. Defaults to TRUE.
legend	Whether to show an interactive legend under the chart (hover to focus a layer, click to hide or show it). Defaults to TRUE.
legendFontSize	Font size of the legend text. Defaults to 11.
valueFormat	A d3-format specifier used to display the values in the tooltip (e.g. ",.0f", "$,.2s", ".1f"). Defaults to ",.2~f".
xFormat	Optional. Format of the x values on the axis and in the tooltip header: a d3-format specifier for numeric x (e.g. "d") or a d3-time-format specifier for dates (e.g. "%b %Y"). Defaults to NULL, which picks a sensible format automatically.
xticks	Optional. Approximate number of x-axis ticks. Defaults to NULL (chosen from the chart width).
xFontSize	Font size of the x-axis tick labels. Defaults to 11.
gridCol	Color of the vertical grid lines drawn at each x tick. Use NA or "transparent" to hide them. Defaults to "#e6e6e6".
xtitle	Optional. Title of the x-axis.
xtitleFontSize	Font size of the x-axis title. Defaults to 13.
title	Optional. Title of the chart.
titleFontSize	Font size of the chart title. Defaults to 16.
font	The font family used for all text. Defaults to "Verdana, Geneva, Tahoma, sans-serif".
bgcol	Background color of the SVG. Defaults to "white".
axisCol	Color of the x-axis, the titles, the crosshair and the legend text. Defaults to "black".
width	Optional. The width of the SVG output.
height	Optional. The height of the SVG output.

Value

An SVG streamgraph.

Examples

```
# Popularity of music genres over four decades (simulated)
set.seed(2026)
years  <- 1980:2020
genres <- c("Rock", "Pop", "Hip hop", "Electronic", "Country",
            "R&B", "Jazz", "Metal", "Reggae", "Latin")
peaks  <- c(1984, 1999, 2012, 2008, 1993, 2003, 1981, 1990, 1986, 2018)
sizes  <- c(60, 55, 70, 40, 30, 35, 18, 25, 15, 45)
music  <- expand.grid(year = years, genre = genres)
idx    <- match(music$genre, genres)
music$listeners <- round(
  sizes[idx] * exp(-((music$year - peaks[idx]) / 9)^2) + runif(nrow(music), 1, 6)
)

streamgraph(
  data   = music,
  x      = "year",
  group  = "genre",
  value  = "listeners",
  valueFormat = ",.0f",
  xtitle = "Year",
  title  = "Music genres, 1980-2020 (millions of listeners)"
)

# Share of the world's telephones by region (base R data), 100% stream
phones <- data.frame(
  year   = rep(as.numeric(rownames(WorldPhones)), ncol(WorldPhones)),
  region = factor(rep(colnames(WorldPhones), each = nrow(WorldPhones)),
                  levels = colnames(WorldPhones)),
  phones = as.vector(WorldPhones)
)

streamgraph(
  data         = phones,
  x            = "year",
  group        = "region",
  value        = "phones",
  offset       = "expand",
  order        = "descending",
  curve        = "monotoneX",
  colorPalette = "Set2",
  valueFormat  = ",.0f",
  title        = "Telephones by region, 1951-1961 (thousands)"
)

# Date x-axis with a silhouette baseline: weekly sign-ups per channel
set.seed(7)
weeks    <- seq(as.Date("2026-01-01"), by = "week", length.out = 52)
```

```
channels <- c("Organic", "Paid search", "Social", "Referral", "Email")
signups  <- expand.grid(week = weeks, channel = channels)
t        <- as.numeric(signups$week - min(weeks)) / 7
k        <- as.integer(signups$channel)
signups$n <- round(
  (60 + 25 * sin(t / 6 + k)) * c(1.4, 1, 0.8, 0.5, 0.6)[k] + rpois(length(t), 6)
)

streamgraph(
  data    = signups,
  x       = "week",
  group   = "channel",
  value   = "n",
  offset  = "silhouette",
  xFormat = "%b %Y",
  colors  = c("#2caffe", "#544fc5", "#00e272", "#fe6a35", "#6b8abc"),
  legend  = FALSE,
  title   = "Weekly sign-ups by channel, 2026"
)
```

`sunburst_chart` *Create a zoomable sunburst chart.*

Description

A sunburst chart displays hierarchical data as concentric rings. The centre represents the whole, each ring is one level of the hierarchy, and every sector's angle is proportional to its value. The look and feel is inspired by the sunburst series of Apache ECharts, reimplemented here in plain D3.

The chart is interactive: hovering a sector highlights it together with its ancestors and shows a tooltip with its full path, value and shares; clicking a sector zooms into it (its children then fill the whole circle with a smooth arc transition) and clicking the centre zooms back out one level.

Usage

```
sunburst_chart(
  data,
  levels,
  value = NULL,
  colors = NULL,
  valueFormat = ",.0f",
  innerRadius = 0.22,
  cornerRadius = 0,
  padAngle = 0,
  strokeCol = "white",
  strokeWidth = 1,
  showLabels = TRUE,
  labelRotate = c("radial", "tangential"),
  labelFontSize = 11,
```

```
  labelCol = NULL,
  rootLabel = "Total",
  centerFontSize = 13,
  zoom = TRUE,
  hover = TRUE,
  animate = TRUE,
  duration = 750,
  title = NULL,
  titleFontSize = 16,
  font = "Verdana, Geneva, Tahoma, sans-serif",
  bgcol = "white",
  textCol = "black",
  width = NULL,
  height = NULL
)
```

Arguments

data	The data frame containing the variables to consider.
levels	A character vector of column names that define the hierarchy, from the outermost category (inner ring) to the most detailed one (outer ring), e.g. `c("continent", "country")`. A missing value (`NA` or `""`) at some level ends that row's path early, which makes ragged hierarchies possible. Rows that share the same path are summed.
value	Optional. The name of a numeric (non-negative) column that determines the size of each sector. Defaults to `NULL`, in which case each row counts as 1 (sector sizes are row counts).
colors	Optional. A vector of colors, one per first-level category (recycled if needed). Deeper levels use progressively lighter shades of their branch color. Defaults to `NULL`, which uses a built-in 10-color palette.
valueFormat	A d3-format specifier used to display values in the centre and the tooltip (e.g. `",.0f"`, `"$,.2s"`, `",.3~s"`). Defaults to `",.0f"`.
innerRadius	Radius of the centre circle as a fraction of the chart's radius, between `0` and `0.9`. The centre shows the current root and its value, and is clicked to zoom out. Defaults to `0.22`.
cornerRadius	Corner radius of the sectors in pixels. Defaults to `0`.
padAngle	Angular padding between sectors in radians (e.g. `0.005`). Defaults to `0`.
strokeCol	Color of the sector borders. Defaults to `"white"`.
strokeWidth	Width of the sector borders in pixels. Defaults to `1`.
showLabels	Whether to display the sector labels. Labels are truncated or hidden automatically when a sector is too small. Defaults to `TRUE`.
labelRotate	Orientation of the sector labels: `"radial"` (default, along the radius) or `"tangential"` (along the arc).
labelFontSize	Font size of the sector labels. Defaults to `11`.

labelCol	Optional. Color of the sector labels. Defaults to NULL, which picks white or near-black automatically for contrast with each sector's fill.
rootLabel	Label shown in the centre (and used as the root name) when the chart is not zoomed. Defaults to "Total".
centerFontSize	Maximum font size of the centre label. Defaults to 13.
zoom	Whether clicking a sector zooms into it and clicking the centre zooms out. Defaults to TRUE.
hover	Whether to highlight the hovered sector (and its ancestors) and show a tooltip. Defaults to TRUE.
animate	Whether the sectors sweep in with an entrance animation. Defaults to TRUE.
duration	Duration of the zoom transition in milliseconds. Defaults to 750.
title	Optional. Title of the chart.
titleFontSize	Font size of the chart title. Defaults to 16.
font	The font family used for all text. Defaults to "Verdana, Geneva, Tahoma, sans-serif".
bgcol	Background color of the SVG. Defaults to "white".
textCol	Color of the title and the centre label. Defaults to "black".
width	Optional. The width of the SVG output.
height	Optional. The height of the SVG output.

Value

An SVG sunburst chart.

Examples

```
# World population in 2007 by continent and country
library(gapminder)

pop_2007 <- gapminder[gapminder$year == 2007, ]

sunburst_chart(
  data        = pop_2007,
  levels      = c("continent", "country"),
  value       = "pop",
  valueFormat = ".3~s",
  rootLabel   = "World",
  title       = "World population in 2007"
)

# A three-level hierarchy: a toy company's headcount
staff <- data.frame(
  division = c(rep("Engineering", 6), rep("Sales", 4), rep("Operations", 4)),
  team     = c("Platform", "Platform", "Data", "Data", "Mobile", "Mobile",
               "EMEA", "EMEA", "Americas", "APAC",
               "Finance", "Finance", "HR", "Legal"),
  role     = c("Backend", "SRE", "Engineers", "Analysts", "iOS", "Android",
```

```
                "Account execs", "Sales engineers", "Account execs",
                "Account execs", "Accounting", "FP&A", "Recruiting", NA),
  people   = c(42, 12, 18, 9, 11, 13, 25, 8, 31, 14, 10, 6, 9, 5)
)

sunburst_chart(
  data        = staff,
  levels      = c("division", "team", "role"),
  value       = "people",
  rootLabel   = "Company",
  labelRotate = "tangential",
  padAngle    = 0.006,
  cornerRadius = 3,
  title       = "Headcount by division, team and role"
)

# Without a value column, sectors are sized by row counts
library(ggplot2) # for the mpg dataset

sunburst_chart(
  data      = mpg,
  levels    = c("class", "manufacturer"),
  rootLabel = "Cars",
  title     = "Number of car models by class and manufacturer"
)
```

`treemap_chart` *Create a treemap chart.*

Description

A treemap displays hierarchical (or flat) part-to-whole data as a set of nested rectangles whose areas are proportional to a value. The technique was invented by Ben Shneiderman in the early 1990s and was made famous by SmartMoney's "Map of the Market" and by the New York Times' graphics of the US federal budget and spending. When a `group` column is supplied the chart becomes a two-level hierarchy: tiles are coloured by group and each group gets its own header and border. Hovering over a tile highlights it and shows a tooltip with its label, group, value and share of the total.

Usage

```
treemap_chart(
  data,
  value,
  label,
  group = NULL,
  tile = "squarify",
  padding = 2,
  colors = NULL,
```

```
  valueFormat = ",.0f",
  showValues = TRUE,
  labelFontSize = 11,
  labelCol = NULL,
  groupFontSize = 12,
  strokeCol = "white",
  strokeWidth = 1,
  hover = TRUE,
  legend = TRUE,
  legendFontSize = 11,
  title = NULL,
  titleFontSize = 16,
  font = "Verdana, Geneva, Tahoma, sans-serif",
  bgcol = "white",
  textCol = "black",
  width = NULL,
  height = NULL
)
```

Arguments

data	The data frame containing the variables to consider.
value	The name of the numeric (non-negative) column that determines the area of each rectangle.
label	The name of the column used as the label of each rectangle (leaf).
group	Optional. The name of a categorical column used as the parent category of each rectangle. When supplied, rectangles are nested inside their group and coloured by group. Defaults to NULL.
tile	The tiling algorithm. One of "squarify" (default), "binary", "slice", "dice" or "sliceDice".
padding	Padding in pixels between rectangles (and around groups). Defaults to 2.
colors	Optional. A vector of colors. When group is supplied, one color is used per group; otherwise one color per rectangle. Colors are recycled if there are fewer colors than groups/rectangles. Defaults to NULL, which uses D3's Tableau 10 palette.
valueFormat	A d3-format specifier used to display values in the labels and tooltip (e.g. ",.0f", "$,.2s", ".3~s"). Defaults to ",.0f".
showValues	Whether to display the formatted value under each label. Defaults to TRUE.
labelFontSize	Font size of the rectangle labels. Labels are automatically truncated or hidden when a rectangle is too small to fit them. Defaults to 11.
labelCol	Optional. Color of the rectangle labels. Defaults to NULL, which picks white or near-black automatically for contrast with each rectangle's fill.
groupFontSize	Font size of the group headers (only used when group is supplied). Defaults to 12.
strokeCol	Color of the rectangle borders. Defaults to "white".

strokeWidth	Width of the rectangle borders in pixels. Defaults to 1.
hover	Whether to highlight the hovered rectangle and show a tooltip. Defaults to TRUE.
legend	Whether to display a legend of the groups (only used when group is supplied). Defaults to TRUE.
legendFontSize	Font size of the legend text. Defaults to 11.
title	Optional. Title of the chart.
titleFontSize	Font size of the chart title. Defaults to 16.
font	The font family used for all text. Defaults to "Verdana, Geneva, Tahoma, sans-serif".
bgcol	Background color of the SVG. Defaults to "white".
textCol	Color of the title and legend text. Defaults to "black".
width	Optional. The width of the SVG output.
height	Optional. The height of the SVG output.

Value

An SVG treemap chart.

Examples

```
# A toy government budget (in millions) by department and ministry
budget <- data.frame(
  department = c("Hospitals", "Primary care", "Public health",
                 "Schools", "Universities", "Vocational training",
                 "Army", "Navy", "Air force",
                 "Roads", "Railways", "Airports",
                 "Pensions", "Unemployment", "Family benefits"),
  ministry   = c("Health", "Health", "Health",
                 "Education", "Education", "Education",
                 "Defence", "Defence", "Defence",
                 "Transport", "Transport", "Transport",
                 "Social security", "Social security", "Social security"),
  spending   = c(5200, 2100, 800,
                 3900, 1800, 600,
                 2300, 1400, 1700,
                 1500, 1100, 400,
                 7400, 1900, 1300)
)

treemap_chart(
  data        = budget,
  value       = "spending",
  label       = "department",
  group       = "ministry",
  valueFormat = "$,.0f",
  title       = "Where the money goes (millions)"
)
```

```
# World population in 2007 by country, grouped by continent
library(gapminder)

pop_2007 <- gapminder[gapminder$year == 2007, ]

treemap_chart(
  data        = pop_2007,
  value       = "pop",
  label       = "country",
  group       = "continent",
  valueFormat = ".3~s",
  padding     = 1,
  title       = "World population in 2007"
)
```

`violin_plot` *Create a violin plot.*

Description

A violin plot shows the distribution of a numeric variable for one or several groups. Each group is drawn as a mirrored kernel density estimate (the "violin"), so the width of the shape at a given value is proportional to how many observations lie around it. Violin plots reveal features that a box plot hides, such as multi-modality or skewness, while an inner box plot keeps the familiar summary statistics in view. The look and feel is inspired by the violin traces of the Plotly.js library.

The kernel density estimates are computed in the browser with a Gaussian kernel. By default each violin gets its own bandwidth, computed with Silverman's rule of thumb (equivalent to `stats::bw.nrd0()`).

The plot is interactive: hovering a violin dims the others, draws a horizontal spike line at the cursor (with its value tagged on the y-axis) and shows a tooltip with the group's summary statistics (n, min, q1, median, q3, max and mean). Hovering a raw point shows its value.

Usage

```
violin_plot(
  data,
  y,
  group = NULL,
  bandwidth = NULL,
  span = c("soft", "hard"),
  scale = c("width", "count"),
  box = TRUE,
  meanline = FALSE,
  points = c("outliers", "all", "none"),
  jitter = 0.6,
  pointRadius = 2.5,
  pointOpacity = 0.7,
  sort = c("none", "mean", "median"),
```

```
  colorPalette = "Category10",
  fillOpacity = 0.5,
  lineWidth = 1.5,
  violinWidth = 0.8,
  hover = TRUE,
  animate = TRUE,
  ytitle = NULL,
  ytitleFontSize = 13,
  xtitle = NULL,
  xtitleFontSize = 13,
  title = NULL,
  titleFontSize = 16,
  xFontSize = 11,
  yFontSize = 11,
  font = "Verdana, Geneva, Tahoma, sans-serif",
  bgcol = "white",
  axisCol = "black",
  gridCol = "#e6e6e6",
  width = NULL,
  height = NULL
)
```

Arguments

data	The data frame containing the variables to consider.
y	The name of the numeric column whose distribution is displayed.
group	Optional. The name of a categorical column used to split the data into one violin per group. When the column is a factor, its level order is used (unless sort is set). Defaults to NULL (a single violin).
bandwidth	Optional. The kernel bandwidth, expressed as the standard deviation of the Gaussian kernel (as in stats::density()), applied to every violin. When NULL (default), each violin uses its own bandwidth computed with Silverman's rule of thumb.
span	How far the density extends. "soft" (default) extends each violin two bandwidths beyond its minimum and maximum so the tails fade out smoothly; "hard" cuts the violin at the observed minimum and maximum.
scale	How the violin widths are scaled. "width" (default) gives every violin the same maximum width; "count" makes the width proportional to the number of observations, so larger groups look larger.
box	Logical. When TRUE (default), an inner box plot is drawn inside each violin: a box from the first to the third quartile, a white median line, and whiskers reaching the most extreme values within 1.5 IQR.
meanline	Logical. When TRUE, a dashed line is drawn across each violin at the mean. Defaults to FALSE.
points	Which raw observations to draw on top of the violins: "outliers" (default, values beyond the whiskers), "all", or "none".

jitter	Horizontal spread of the points, as a fraction (0 to 1) of the violin's width at the point's value. Points therefore always stay inside their violin. Defaults to 0.6. Use 0 to align them on the center line.
pointRadius	Radius of the points in pixels. Defaults to 2.5.
pointOpacity	Fill opacity of the points (0 to 1). Defaults to 0.7.
sort	How to order the violins from left to right. One of "none" (default, factor level order or order of appearance), "mean", or "median" (ascending).
colorPalette	The violin colors. Either the name of a D3 categorical color scheme (e.g. "Category10", the default, which is also Plotly's classic colorway, "Tableau10", "Set2"), a single color applied to every violin, an unnamed vector of colors recycled over the groups (in their original order), or a named vector whose names match the group values.
fillOpacity	Fill opacity of the violins (0 to 1). Defaults to 0.5.
lineWidth	Width of the violin outline in pixels. Defaults to 1.5.
violinWidth	Maximum width of a violin, as a fraction (0 to 1] of the space allotted to each group. Defaults to 0.8.
hover	Logical. When TRUE (default), hovering a violin highlights it and shows a tooltip with its summary statistics.
animate	Logical. When TRUE (default), the violins grow from their center line and the points fade in on first render.
ytitle	Optional. Title of the y-axis.
ytitleFontSize	Font size of the y-axis title. Defaults to 13.
xtitle	Optional. Title of the x-axis.
xtitleFontSize	Font size of the x-axis title. Defaults to 13.
title	Optional. Title of the chart.
titleFontSize	Font size of the chart title. Defaults to 16.
xFontSize	Font size of the group labels. Defaults to 11. Labels are rotated automatically when they do not fit.
yFontSize	Font size of the y-axis tick labels. Defaults to 11.
font	The font family used for all text. Defaults to "Verdana, Geneva, Tahoma, sans-serif".
bgcol	Background color of the SVG. Defaults to "white".
axisCol	Color of axis lines, ticks, and labels. Defaults to "black".
gridCol	Color of the horizontal grid lines. Defaults to "#e6e6e6". Use "none" to hide them.
width	Optional. The width of the SVG output.
height	Optional. The height of the SVG output.

Value

An SVG violin plot.

Examples

```
# Sepal length by species, with the inner box plot and outliers
violin_plot(
  data   = iris,
  y      = "Sepal.Length",
  group  = "Species",
  title  = "Sepal length by species",
  ytitle = "Sepal length (cm)"
)

# Every observation jittered inside its violin, plus a mean line
library(ggplot2) # for the mpg dataset

violin_plot(
  data         = mpg,
  y            = "hwy",
  group        = "class",
  points       = "all",
  meanline     = TRUE,
  sort         = "median",
  colorPalette = "Tableau10",
  title        = "Highway fuel economy by vehicle class",
  ytitle       = "Highway miles per gallon"
)

# A bimodal distribution, width proportional to group size, hard span
set.seed(2026)
waiting <- data.frame(
  minutes = c(rnorm(300, 55, 6), rnorm(200, 80, 5),
              rnorm(150, 70, 10), rnorm(60, 62, 4)),
  station = rep(c("North", "Central", "South"), c(500, 150, 60))
)

violin_plot(
  data         = waiting,
  y            = "minutes",
  group        = "station",
  scale        = "count",
  span         = "hard",
  colorPalette = c(North = "#636efa", Central = "#ef553b",
                   South = "#00cc96"),
  title        = "Waiting time by station",
  ytitle       = "Minutes",
  xtitle       = "Station"
)

# A single violin with a custom bandwidth
violin_plot(
  data         = faithful,
  y            = "eruptions",
  bandwidth    = 0.2,
  colorPalette = "#9467bd",
```

```
  points       = "all",
  ytitle       = "Eruption time (min)"
)
```

`waffle_chart` *Create a waffle chart.*

Description

A waffle chart is a grid of squares (by default 10 x 10 = 100 cells) in which each category fills a number of cells proportional to its share of the whole. Each cell therefore reads as "one percent" (or one unit of the total), which makes part-to-whole comparisons far easier to read than a pie chart. The format is a staple of Pew Research Center, FiveThirtyEight, The Economist and the Washington Post, and was popularised in R by Bob Rudis's `waffle` package.

Cells are allocated with the largest-remainder method, so the cell counts always add up exactly to `rows * cols`. Unlike `parliament_chart`, which draws individual seats along a hemicycle, a waffle chart uses a rectangular grid to show proportions of a whole rather than absolute counts.

Usage

```
waffle_chart(
  data,
  category,
  value,
  rows = 10,
  cols = 10,
  total = NULL,
  colors = NULL,
  emptyCol = "#e5e5e5",
  cellPadding = 2,
  cellRadius = 2,
  direction = "row",
  legend = TRUE,
  legendFontSize = 12,
  hover = TRUE,
  animate = TRUE,
  animationDuration = 1200,
  title = NULL,
  titleFontSize = 16,
  font = "Verdana, Geneva, Tahoma, sans-serif",
  bgcol = "white",
  textCol = "black",
  width = NULL,
  height = NULL
)
```

Arguments

data	The data frame containing the variables to consider. It should contain one row per category.
category	The name of the categorical column used as labels.
value	The name of the numeric (non-negative) column holding each category's value.
rows	Number of rows in the grid. Defaults to 10.
cols	Number of columns in the grid. Defaults to 10. The total number of cells is rows * cols.
total	Optional. The value represented by the whole grid. Defaults to the sum of value. When larger than that sum, the remaining cells are left unfilled and drawn with emptyCol.
colors	Optional. A character vector of colors, one per category (recycled if shorter). Defaults to a Tableau-10 style palette.
emptyCol	Color of the unfilled cells. Defaults to "#e5e5e5".
cellPadding	Gap between cells in pixels. Defaults to 2.
cellRadius	Corner radius of each cell in pixels. Defaults to 2. Use 0 for sharp squares.
direction	Fill direction. "row" (default) fills row by row starting from the bottom-left corner (the Pew Research style), "col" fills column by column starting from the bottom-left corner.
legend	Whether to display a legend showing each category and its percentage. Defaults to TRUE.
legendFontSize	Font size of the legend text. Defaults to 12.
hover	Whether hovering a cell or a legend entry highlights that category and shows a label with its value and percentage. Defaults to TRUE.
animate	Whether the cells pop in sequentially when the chart loads. Defaults to TRUE.
animationDuration	Total duration of the load animation in milliseconds. Defaults to 1200.
title	Optional. Title of the chart.
titleFontSize	Font size of the chart title. Defaults to 16.
font	The font family used for all text. Defaults to "Verdana, Geneva, Tahoma, sans-serif".
bgcol	Background color of the SVG. Defaults to "white".
textCol	Color of the title and legend text. Defaults to "black".
width	Optional. The width of the SVG output.
height	Optional. The height of the SVG output.

Value

An SVG waffle chart.

Examples

```
# How respondents answered a survey question
survey <- data.frame(
  response = c("Strongly agree", "Agree", "Neutral",
               "Disagree", "Strongly disagree"),
  n = c(312, 405, 158, 87, 38)
)

waffle_chart(
  data     = survey,
  category = "response",
  value    = "n",
  colors   = c("#1a9641", "#a6d96a", "#bdbdbd", "#fdae61", "#d7191c"),
  title    = "Do you support the new policy?"
)

# Share of the world population by continent in 2007, filled by column
library(gapminder)

pop_2007 <- gapminder[gapminder$year == 2007, ]
pop_continent <- aggregate(pop ~ continent, data = pop_2007, FUN = sum)

waffle_chart(
  data      = pop_continent,
  category  = "continent",
  value     = "pop",
  direction = "col",
  rows      = 10,
  cols      = 20,
  title     = "World population by continent, 2007"
)
```

`waterfall_chart` *Create an interactive waterfall chart.*

Description

A waterfall chart shows how an initial value is built up or eroded by a series of positive and negative contributions to reach a final total. Each bar starts exactly where the previous one ended, making cumulative effects immediately visible. It is the standard chart for P&L decomposition, budget variance analysis, and any "what changed and why" narrative.

Hovering over a bar shows a tooltip with the label, the delta value, and the running total at that point.

Usage

```
waterfall_chart(
  data,
  x,
```

```
  y,
  measure = NULL,
  positiveColor = "#4CAF50",
  negativeColor = "#F44336",
  totalColor = "#5C85D6",
  connector = TRUE,
  connectorColor = "#aaaaaa",
  showLabels = TRUE,
  labelFontSize = 10,
  title = NULL,
  titleFontSize = 16,
  xtitle = NULL,
  xtitleFontSize = 13,
  ytitle = NULL,
  ytitleFontSize = 13,
  xFontSize = 11,
  yFontSize = 11,
  opacity = 1,
  font = "Verdana, Geneva, Tahoma, sans-serif",
  bgcol = "white",
  axisCol = "black",
  width = NULL,
  height = NULL
)
```

Arguments

`data`	The data frame containing the variables to consider.
`x`	The name of the column containing category labels.
`y`	The name of the column containing the delta values (positive for increases, negative for decreases).
`measure`	Optional. The name of a column that classifies each row as `"relative"` (default — a delta added to the running total) or `"total"` (a bar spanning from zero to the current running total, used for subtotals and the final total).
`positiveColor`	Fill color for bars representing an increase. Defaults to `"#4CAF50"`.
`negativeColor`	Fill color for bars representing a decrease. Defaults to `"#F44336"`.
`totalColor`	Fill color for `"total"` measure bars. Defaults to `"#5C85D6"`.
`connector`	Logical. Whether to draw dashed connector lines between consecutive bars. Defaults to `TRUE`.
`connectorColor`	Color of the connector lines. Defaults to `"#aaaaaa"`.
`showLabels`	Logical. Whether to display the delta value above or below each bar. Defaults to `TRUE`.
`labelFontSize`	Font size of the bar value labels. Defaults to `10`.
`title`	Optional. Title of the chart.
`titleFontSize`	Font size of the chart title. Defaults to `16`.

xtitle	Optional. Title of the x-axis.
xtitleFontSize	Font size of the x-axis title. Defaults to 13.
ytitle	Optional. Title of the y-axis.
ytitleFontSize	Font size of the y-axis title. Defaults to 13.
xFontSize	Font size of the x-axis tick labels. Defaults to 11.
yFontSize	Font size of the y-axis tick labels. Defaults to 11.
opacity	Opacity of the bars (0 to 1). Defaults to 1.
font	The font family used for all text. Defaults to "Verdana, Geneva, Tahoma, sans-serif".
bgcol	Background color of the SVG. Defaults to "white".
axisCol	Color of axis lines, ticks, and labels. Defaults to "black".
width	Optional. The width of the SVG output.
height	Optional. The height of the SVG output.

Value

An interactive SVG waterfall chart.

Examples

```
# P&L bridge: from revenue down to net income
pnl <- data.frame(
  label   = c("Revenue", "COGS", "Gross Profit",
              "R&D", "S&M", "G&A", "Operating Income"),
  value   = c(1200, -450, 750, -120, -90, -60, 480),
  measure = c("relative", "relative", "total",
              "relative", "relative", "relative", "total")
)

waterfall_chart(
  data    = pnl,
  x       = "label",
  y       = "value",
  measure = "measure",
  title   = "P&L bridge",
  ytitle  = "USD thousands"
)

# Monthly cash flow
cashflow <- data.frame(
  month = c("Jan", "Feb", "Mar", "Apr", "May", "Jun"),
  delta = c(120, -30, 80, -60, 95, -20)
)

waterfall_chart(
  data           = cashflow,
  x              = "month",
  y              = "delta",
```

```
  title          = "Monthly cash flow",
  ytitle         = "USD thousands",
  positiveColor  = "steelblue",
  negativeColor  = "tomato"
)
```

Index

anim_line_chart, 4
animated_histogram, 3
area_band, 6
area_chart, 8

bar_chart, 10
bar_chart_race, 12
beeswarm_plot, 14
bullet_chart, 17
bump_chart, 19

calendar_heatmap, 21
chord_diagram, 25

dumbbell_chart, 28

flame, 31
flower, 32

gauge_chart, 33
glass_fill, 35

heart_fill, 36
hexbin_plot, 38
histogram, 41
horz_bar_chart, 43
horz_lollipop, 45

line_chart, 46
liquid_chart, 52
lollipop_chart, 53

mosaic_chart, 55

parliament_chart, 59, *98*
pie_chart, 60
plant_growth, 62
pulse_grid, 63

radar_chart, 65
radial_bar_chart, 68
ridgeline_plot, 71
rose_chart, 74

scatter_plot, 77
slope_chart, 79
stacked_area_chart, 82, *84*
streamgraph, 84
sunburst_chart, 88

treemap_chart, 91

violin_plot, 94

waffle_chart, 98
waterfall_chart, 100