Package 'vchartr'

May 8, 2026

Title Interactive Charts with the 'JavaScript' 'VChart' Library

Version 0.1.5

Description Provides an 'htmlwidgets' interface to 'VChart.js'.
'VChart', more than just a cross-platform charting library, but also an expressive data storyteller.
'VChart' examples and documentation are available here: <https://www.visactor.io/vchart>.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.3

Imports ggplot2, htmlwidgets, magrittr, rlang, scales

Suggests bslib, knitr, geojsonio, palmerpenguins, rmarkdown, sf, shiny

VignetteBuilder knitr

Depends R (>= 2.10)

LazyData true

URL https://dreamrs.github.io/vchartr/

NeedsCompilation no

Author Victor Perrier [aut, cre],
Fanny Meyer [aut]

Maintainer Victor Perrier <victor.perrier@dreamrs.fr>

Repository CRAN

Date/Publication 2026-01-07 21:30:02 UTC

Contents

`vchartr-package` *An* `htmlwidget` *interface to the VChart javascript chart library*

Description

This package allow you to use VChart.js (https://www.visactor.io/vchart), to create interactive charts.

Author(s)

Victor Perrier (@dreamRs_fr)

See Also

Useful links:

- https://dreamrs.github.io/vchartr/

`co2_emissions` *CO2 emissions*

Description

This dataset represents CO2 emissions for a subset of country over the period 1990 - 2022.

Usage

```
co2_emissions
```

Format

A data frame with 495 observations and 11 variables:

- `country` : Country - Geographic location.
- `year` : Year - Year of observation.
- `co2` : Annual CO2 emissions - Annual total emissions of carbon dioxide (CO2), excluding land-use change, measured in million tonnes.

- co2_per_gdp : Annual CO2 emissions per GDP (kg per international-$) - Annual total emissions of carbon dioxide (CO2), excluding land-use change, measured in kilograms per dollar of GDP (2011 international-$).
- co2_per_capita : Annual CO2 emissions (per capita) - Annual total emissions of carbon dioxide (CO2), excluding land-use change, measured in tonnes per person.
- co2_growth_abs : Annual CO2 emissions growth (abs) - Annual growth in total emissions of carbon dioxide (CO2), excluding land-use change, measured in million tonnes.
- co2_growth_prct : Annual CO2 emissions growth (%) - Annual percentage growth in total emissions of carbon dioxide (CO2), excluding land-use change.
- co2_per_unit_energy : Annual CO2 emissions per unit energy (kg per kilowatt-hour) - Annual total emissions of carbon dioxide (CO2), excluding land-use change, measured in kilograms per kilowatt-hour of primary energy consumption.
- consumption_co2 : Annual consumption-based CO2 emissions - Annual consumption-based emissions of carbon dioxide (CO2), measured in million tonnes.
- consumption_co2_per_capita : Per capita consumption-based CO2 emissions - Annual consumption-based emissions of carbon dioxide (CO2), measured in tonnes per person.
- consumption_co2_per_gdp : Annual consumption-based CO2 emissions per GDP (kg per international-$) - Annual consumption-based emissions of carbon dioxide (CO2), measured in kilograms per dollar of GDP (2011 international-$).

Note

Documentation is from Our World In Data, see https://github.com/owid/co2-data for the data and https://ourworldindata.org/co2-and-greenhouse-gas-emissions for more about CO2 emissions.

Source

Our World In Data

---

co2_world *World CO2 emissions*

Description

This dataset contains world polygons with CO2 emissions.

Usage

```
co2_world
```

Format

A data frame with 495 observations and 11 variables:

- `iso_code` : ISO code A3 for country.
- `name` : Name of country.
- `co2` : Annual CO2 emissions - Annual total emissions of carbon dioxide (CO2), excluding land-use change, measured in million tonnes.
- `co2_per_capita` : Annual CO2 emissions (per capita) - Annual total emissions of carbon dioxide (CO2), excluding land-use change, measured in tonnes per person.
- `geometry` : Geographical attributes.

Note

Documentation is from Our World In Data, see https://github.com/owid/co2-data for the data and https://ourworldindata.org/co2-and-greenhouse-gas-emissions for more about CO2 emissions.

Source

Our World In Data

---

countries_gdp *Countries GDP*

Description

These data represent the GDP of the world's countries, classified by continent and sub-region. This is a subset of the dataset `rnaturalearth::countries110`.

Usage

```
countries_gdp
```

Format

A data frame with 177 observations and 3 variables:

- `REGION_UN` : Continent
- `SUBREGION` : Sub-region in the continent
- `ADMIN` : Administrative name of country
- `GDP_MD` : GDP

Source

Package rnaturalearth

eco2mix — *Monthly electricity generation by source in France*

Description

This dataset represents monthly electricity generation by source in France over the period 2012 - 2024.

Usage

```
eco2mix
```

Format

A data frame with 151 observations and 10 variables:

- `date` : Date
- `fuel` : Fuel generation in MW
- `coal` : Coal generation in MW
- `gas` : Gas generation in MW
- `nuclear` : Nuclear generation in MW
- `wind` : Wind generation in MW
- `solar` : Solar generation in MW
- `hydraulic` : Hydraulic generation in MW
- `pumping` : Pumping generation in MW
- `bioenergies` : Bioenergies generation in MW

Source

eco2mix

eco2mix_long — *Monthly electricity generation by source in France (long format)*

Description

This dataset represents monthly electricity generation by source in France over the period 2012 - 2024.

Usage

```
eco2mix_long
```

Format

A data frame with 1359 observations and 3 variables:

- `date` : Date
- `source` : Production according to the different sectors making up the energy mix.
- `production` : Generation in MW

Source

eco2mix

`electricity_mix` *Electricity mix for 10 countries*

Description

This dataset represents the electricity mix of 10 countries (those with the highest electricity generation) in 2023.

Usage

```
electricity_mix
```

Format

A data frame with 70 observations and 3 variables:

- `country` : Country name
- `source` : source of electricity
- `generation` : Total electricity generation - Measured in terawatt-hours.
- `type` : Low carbon or fossil fuels type of source.

Source

Our World In Data

energy_sankey *Data for Sankey Chart*

Description

These data represent how energy is converted or transmitted before being consumed or lost.

Usage

```
energy_sankey
```

Format

A data frame with 177 observations and 3 variables:

- `source` : Source
- `target` : Target
- `value` : Energy in TWh

Source

Department of Energy & Climate Change via Tom Counsell

format-date *Format date with dayjs JavaScript library*

Description

Format date with dayjs JavaScript library

Usage

```
format_date_dayjs(format, prefix = "", suffix = "", locale = "en")

format_datetime_dayjs(
  format,
  prefix = "",
  suffix = "",
  locale = "en",
  tz = NULL
)

label_format_date(format)

label_format_datetime(format, tz = NULL)
```

Arguments

format	Format for dates, see online documentation.
prefix	Character string to append before formatted value.
suffix	Character string to append after formatted value.
locale	Localization to use, for example "fr" for french, see possible values online.
tz	Timezone to use.

Value

a JS function.

Examples

```
library(vchartr)



### Format date

# date in french in %B %y format
vchart(eco2mix) %>%
  v_line(aes(date, solar)) %>%
  v_scale_x_date(
    date_labels = format_date_dayjs("MMMM YY", locale = "fr")
  )

# date in arabic in %A %d %b %Y format
vchart(eco2mix) %>%
  v_line(aes(date, solar)) %>%
  v_scale_x_date(
    date_labels = format_date_dayjs("dddd D MMM YYYY", locale = "ar")
  )
```

---

format_num_d3 *Format numbers with D3*

Description

Format numbers with D3

Usage

```
format_num_d3(format, prefix = "", suffix = "", locale = "en-US")
```

Arguments

format	Format for numbers, currency, percentage, e.g. ".0\%" for rounded percentage. See online documentation : https://github.com/d3/d3-format.
prefix	Character string to append before formatted value.
suffix	Character string to append after formatted value.
locale	Localization to use, for example "fr-FR" for french, see possible values here: https://github.com/d3/d3-format/tree/master/locale.

Value

a JS function.

Examples

```
library(vchartr)
```

---

mark-area *Add a rectangle annotation to a chart*

Description

Add a rectangle annotation to a chart

Usage

```
v_mark_rect(
  vc,
  xmin = NULL,
  xmax = NULL,
  ymin = NULL,
  ymax = NULL,
  .area.style.fill = "grey35",
  .area.style.fillOpacity = 0.3,
  .label.text = NULL,
  .label.position = "insideTop",
  .label.refY = 0,
  .label.refX = 0
)

v_mark_polygon(
  vc,
  coords,
  .area.style.fill = "grey35",
  .area.style.fillOpacity = 0.3,
  .label.text = NULL,
```

```
  .label.position = "insideTop",
  .label.refY = 0,
  .label.refX = 0
)
```

Arguments

vc	An htmlwidget created with vchart().
xmin, xmax, ymin, ymax	Target position for the rectangle. Use NULL to target chart's limits. You can also use relative values, e.g. "50%".
.area.style.fill	Fill color.
.area.style.fillOpacity	Fill opacity.
.label.text	Text for the label on the line.
.label.position	The label position of the dimension line (the relative position of the label relative to the line). See online documentation for options.
.label.refY, .label.refX	The offset in the vertical direction of the reference line.
coords	A data.frame (or something that can be converted to data.frame) with two columns, first will be used as x coordinates, second as y.

Value

A vchart() htmlwidget object.

Examples

```
library(vchartr)

# Draw a rectangle
vchart(cars) %>%
  v_scatter(aes(speed, dist)) %>%
  v_mark_rect(
    xmin = 10,
    xmax = 18,
    ymin = 20,
    ymax = 50
  )

# don't provide x or y to reach chart's limit
vchart(cars) %>%
  v_scatter(aes(speed, dist)) %>%
  v_mark_rect(
    xmin = 10,
    xmax = 18
  )
```

```
vchart(cars) %>%
  v_scatter(aes(speed, dist)) %>%
  v_mark_rect(
    ymin = 10,
    ymax = 18
  )


vchart(cars) %>%
  v_scatter(aes(speed, dist)) %>%
  v_mark_rect(
    xmin = "50%",
    xmax = "100%", # from right to left
    ymin = "50%",
    ymax = "100%" # note that for y it's from top to bottom
  )


# Whith date scale
vchart(temperatures) %>%
  v_line(aes(date, average)) %>%
  v_mark_rect(
    xmin = as.Date("2024-06-20"),
    xmax = as.Date("2024-09-22"),
    .label.text = "Summer"
  )


# Draw a polygon
vchart(cars) %>%
  v_scatter(aes(speed, dist)) %>%
  v_mark_polygon(
    coords = list(
      x = c(7, 22, 15),
      y = c(10, 50, 80)
    )
  )
```

`mark-line` *Add an horizontal or vertical line to a chart*

Description

Add an horizontal or vertical line to a chart

Usage

```
v_mark_vline(
  vc,
```

```
  x,
  ...,
  .line.style.stroke = "#000",
  .line.style.lineDash = list(8, 8),
  .label.text = NULL,
  .label.position = "end",
  .label.refY = 0,
  .label.refX = 0,
  .endSymbol.style.visible = FALSE,
  .startSymbol.style.visible = FALSE
)

v_mark_hline(
  vc,
  y,
  ...,
  .line.style.stroke = "#000",
  .line.style.lineDash = list(8, 8),
  .label.text = NULL,
  .label.position = "insideEndBottom",
  .label.refY = -10,
  .label.refX = 0,
  .endSymbol.style.visible = FALSE,
  .startSymbol.style.visible = FALSE
)

v_mark_segment(
  vc,
  x,
  xend,
  y,
  yend,
  ...,
  .line.style.stroke = "#000",
  .line.style.lineDash = list(8, 8),
  .label.text = NULL,
  .label.position = "insideEndBottom",
  .label.refY = -10,
  .label.refX = 0,
  .endSymbol.style.visible = FALSE,
  .startSymbol.style.visible = FALSE
)
```

Arguments

vc	An htmlwidget created with vchart().
x, y, xend, yend	Target position for the line.
...	Additional parameters for the line, see online documentation for more.

`.line.style.stroke`
: Stroke color.

`.line.style.lineDash`
: Used to configure the dashed line mode when filling lines. It uses an array of values to specify the alternating lengths of lines and gaps that describe the pattern.

`.label.text`
: Text for the label on the line.

`.label.position`
: The label position of the dimension line (the relative position of the label relative to the line). See online documentation for options.

`.label.refY`, `.label.refX`
: The offset in the vertical direction of the reference line.

`.endSymbol.style.visible`, `.startSymbol.style.visible`
: Whether the symbol element is visible or not.

Value

A `vchart()` `htmlwidget` object.

Examples

```
library(vchartr)

# Vertical line
vchart(meteo_paris) %>%
  v_line(aes(month, temperature_avg)) %>%
  v_mark_vline(x = "May")

# Vertical lines with labels
vchart(meteo_paris) %>%
  v_line(aes(month, temperature_avg)) %>%
  v_mark_vline(
    x = c("May", "September"),
    .label.text = c("May", "September")
  )

# Horizontal line
vchart(meteo_paris) %>%
  v_line(aes(month, temperature_avg)) %>%
  v_mark_hline(y = 12)

# Both horizontal and vertical lines
vchart(meteo_paris) %>%
  v_line(aes(month, temperature_avg)) %>%
  v_mark_vline(x = "May") %>%
  v_mark_hline(y = 12)

# lines on a scatter plot
vchart(cars) %>%
  v_scatter(aes(speed, dist)) %>%
  v_mark_vline(x = mean(cars$speed)) %>%
```

```
  v_mark_hline(y = mean(cars$dist))

# segment
vchart(cars) %>%
  v_scatter(aes(speed, dist)) %>%
  v_mark_segment(x = 8, xend = 22, y = 12, yend = 100)

# line on date scale
vchart(temperatures) %>%
  v_line(aes(date, average)) %>%
  v_mark_vline(x = as.Date("2024-06-20"))

# segment on date scale
vchart(temperatures) %>%
  v_line(aes(date, average)) %>%
  v_mark_segment(
    x = as.Date("2024-04-01"), xend = as.Date("2024-07-01"),
    y = 12, yend = 24,
    .line.style.lineDash = 0,
    .line.style.stroke = "firebrick"
  )
```

`meteo_paris` *Paris climate*

Description

This data contains information about the climate in Paris, France.

Usage

```
meteo_paris
```

Format

A data frame with 177 observations and 3 variables:

- `month` : Month of the year
- `temperature_avg` : Average temperature (°C)
- `temperature_min` : Average minimum temperature (°C)
- `temperature_max` : Average maximum temperature (°C)
- `precipitation` : Precipitation (mm)
- `humidity` : Humidity (%)
- `rainy_days` : Rainy days (days)
- `sunshine_hours` : Sunshine hours (h)

Source

Climate-data.org

`scale-color-manual` *Manual color scale*

Description

Manual color scale

Discrete color scale

Usage

```
v_scale_color_manual(vc, values)

v_scale_fill_manual(vc, values)

v_scale_color_discrete(vc, palette)

v_scale_fill_discrete(vc, palette)
```

Arguments

`vc`	An htmlwidget created with `vchart()` or specific chart's type function.
`values`	A named list with data values as name and color as values
`palette`	A color vector or the name of an R palette.

Value

A `vchart()` `htmlwidget` object.

A `vchart()` `htmlwidget` object.

Examples

```
library(vchartr)

subset(electricity_mix, country %in% c("France", "Canada")) %>%
  vchart() %>%
  v_bar(aes(country, generation, fill = source)) %>%
  v_scale_fill_manual(c(
    "oil" = "#80549f",
    "coal" = "#a68832",
    "solar" = "#d66b0d",
    "gas" = "#f20809",
    "wind" = "#72cbb7",
    "hydro" = "#2672b0",
    "nuclear" = "#e4a701"
```

```
  ))

vchart(palmerpenguins::penguins) %>%
  v_scatter(
    aes(x = flipper_length_mm, y = body_mass_g, color = species)
  ) %>%
  v_scale_color_manual(c(
    Adelie = "#ffa232",
    Chinstrap = "#33a2a2",
    Gentoo = "#b34df2"
  ))

library(vchartr)

subset(electricity_mix, country %in% c("France", "Canada")) %>%
  vchart() %>%
  v_bar(aes(country, generation, fill = source)) %>%
  v_scale_fill_discrete("Okabe-Ito")

subset(electricity_mix, country %in% c("France", "Canada")) %>%
  vchart() %>%
  v_bar(aes(country, generation, fill = source)) %>%
  v_scale_fill_discrete("ggplot2")

# or
subset(electricity_mix, country %in% c("France", "Canada")) %>%
  vchart() %>%
  v_bar(aes(country, generation, fill = source)) %>%
  v_scale_fill_discrete(palette.colors(palette = "ggplot2")[-1])
```

`scale-continuous` *Axis scale for continuous data*

Description

Axis scale for continuous data

Usage

```
v_scale_x_continuous(
  vc,
  name = NULL,
  breaks = NULL,
  pretty = TRUE,
  labels = NULL,
  labels_tooltip = labels,
  zero = NULL,
  min = NULL,
  max = NULL,
```

```
  ...,
  position = "bottom"
)

v_scale_y_continuous(
  vc,
  name = NULL,
  breaks = NULL,
  pretty = TRUE,
  labels = NULL,
  labels_tooltip = labels,
  zero = NULL,
  min = NULL,
  max = NULL,
  ...,
  position = "left"
)

v_scale_x_log(
  vc,
  name = NULL,
  breaks = NULL,
  pretty = TRUE,
  labels = NULL,
  labels_tooltip = labels,
  zero = NULL,
  min = NULL,
  max = NULL,
  ...,
  position = "bottom"
)

v_scale_y_log(
  vc,
  name = NULL,
  breaks = NULL,
  pretty = TRUE,
  labels = NULL,
  labels_tooltip = labels,
  zero = NULL,
  min = NULL,
  max = NULL,
  ...,
  position = "left"
)
```

Arguments

vc	An htmlwidget created with `vchart()` or specific chart's type function.

name	Title for the axis.
breaks	One of: • A single numeric value giving the number of breaks. • A numeric vector of positions.
pretty	Use pretty() to identify breaks if breaks is a single numeric value.
labels, labels_tooltip	The format to be applied on numeric in the labels/tooltip. Either: • A single character indicating the D3 format. • A JS function, such as format_num_d3().
zero	Force axis to start at 0.
min	Minimum value on the axis.
max	Maximum value on the axis.
...	Additional parameters for the axis.
position	Position of the axis.

Value

A vchart() htmlwidget object.

Examples

```
library(vchartr)

# Add a title to the axis
vchart(top_generation) %>%
  v_bar(aes(country, electricity_generation)) %>%
  v_scale_y_continuous(name = "Electricity generation")

vchart(subset(world_electricity, type == "total")) %>%
  v_bar(aes(year, generation, fill = source)) %>%
  v_scale_y_continuous(name = "Electricity generation")

# Specify number of breaks
vchart(top_generation) %>%
  v_bar(aes(country, electricity_generation)) %>%
  v_scale_y_continuous(breaks = 10)

# Specify breaks position
vchart(top_generation) %>%
  v_bar(aes(country, electricity_generation)) %>%
  v_scale_y_continuous(breaks = c(0, 5000, 10000))

# Format labels
vchart(top_generation) %>%
  v_bar(aes(country, electricity_generation)) %>%
  v_scale_y_continuous(labels = "~s")

# Format labels with options
```

```
vchart(top_generation) %>%
  v_bar(aes(country, electricity_generation)) %>%
  v_scale_y_continuous(labels = format_num_d3(",", suffix = " TWh", locale = "fr-FR"))

vchart(subset(world_electricity, type == "total")) %>%
  v_bar(aes(year, generation, fill = source)) %>%
  v_scale_y_continuous(labels = format_num_d3(",", suffix = " TWh", locale = "fr-FR"))
```

---

scale-date *Axis scale for date/time data*

Description

Axis scale for date/time data

Usage

```
v_scale_x_date(
  vc,
  name = NULL,
  date_breaks = NULL,
  date_labels = NULL,
  date_labels_tooltip = date_labels,
  min = NULL,
  max = NULL,
  ...,
  position = "bottom"
)

v_scale_y_date(
  vc,
  name = NULL,
  date_breaks = NULL,
  date_labels = NULL,
  date_labels_tooltip = date_labels,
  min = NULL,
  max = NULL,
  ...,
  position = "left"
)

v_scale_x_datetime(
  vc,
  name = NULL,
  date_breaks = NULL,
  date_labels = NULL,
  date_labels_tooltip = date_labels,
```

```
  tz = NULL,
  min = NULL,
  max = NULL,
  ...,
  position = "bottom"
)

v_scale_y_datetime(
  vc,
  name = NULL,
  date_breaks = NULL,
  date_labels = NULL,
  date_labels_tooltip = date_labels,
  tz = NULL,
  min = NULL,
  max = NULL,
  ...,
  position = "left"
)
```

Arguments

vc	An htmlwidget created with vchart() or specific chart's type function.
name	Title for the axis.
date_breaks	One of: • A single numeric value giving the number of breaks. • A string giving the distance between breaks like "2 weeks", or "10 years". • A Date/POSIXct vector giving positions of breaks.
date_labels	The format to be applied on Date/POSIXct in the labels, see format_date_dayjs().
date_labels_tooltip	The format to be applied on Date/POSIXct in the tooltip, see format_date_dayjs().
min	Minimum value on the axis.
max	Maximum value on the axis.
...	Additional parameters for the axis.
position	Position of the axis.
tz	The timezone.

Value

A vchart() htmlwidget object.

Examples

```
library(vchartr)

# Add a title to the axis
```

```
vchart(eco2mix) %>%
  v_line(aes(date, solar)) %>%
  v_scale_x_date(name = "Date")

# Specify number of labels
vchart(eco2mix) %>%
  v_line(aes(date, solar)) %>%
  v_scale_x_date(date_breaks = 5)

# Specify intervals between labels
vchart(eco2mix) %>%
  v_line(aes(date, solar)) %>%
  v_scale_x_date(date_breaks = "2 years")

# Format labels
vchart(eco2mix) %>%
  v_line(aes(date, solar)) %>%
  v_scale_x_date(date_labels = "MM-YYYY")

# Other format for labels
vchart(eco2mix) %>%
  v_line(aes(date, solar)) %>%
  v_scale_x_date(date_labels = "MMM YYYY")

# Format labels with locale
vchart(eco2mix) %>%
  v_line(aes(date, solar)) %>%
  v_scale_x_date(
    date_labels = format_date_dayjs("MMMM YY", locale = "fr")
  )

# Different formats in labels and tootlip
vchart(eco2mix) %>%
  v_line(aes(date, solar)) %>%
  v_scale_x_date(
    date_labels = "YYYY-MM",
    date_labels_tooltip = "MMMM YYYY"
  )
```

`scale-discrete` *Axis scale for discrete data*

Description

Axis scale for discrete data

Usage

```
v_scale_x_discrete(vc, name = NULL, ..., position = "bottom")
```

```
v_scale_y_discrete(vc, name = NULL, ..., position = "left")
```

Arguments

vc	An htmlwidget created with vchart() or specific chart's type function.
name	Title for the axis.
...	Additional parameters for the axis.
position	Position of the axis.

Value

A vchart() htmlwidget object.

Examples

```
library(vchartr)
```

---

scale-gradient *Color scale for continuous data*

Description

Color scale for continuous data

Usage

```
v_scale_colour_gradient(
  vc,
  name = NULL,
  low = "#132B43",
  high = "#56B1F7",
  limits = NULL,
  position = c("right", "bottom", "left", "top"),
  align = c("middle", "start", "end")
)

v_scale_fill_gradient(
  vc,
  name = NULL,
  low = "#132B43",
  high = "#56B1F7",
  limits = NULL,
  position = c("right", "bottom", "left", "top"),
  align = c("middle", "start", "end")
)
```

Arguments

vc	An htmlwidget created with vchart() or specific chart's type function.
name	Title for the legend.
low, high	Colours for low and high ends of the gradient.
limits	Limits of the scale, default (NULL) is to use the default scale range of the data.
position	Position of the legend.
align	Alignment of the legend.

Value

A vchart() htmlwidget object.

Examples

```
library(vchartr)
data("penguins", package = "palmerpenguins")

vchart(penguins) %>%
  v_scatter(aes(
    x = bill_length_mm,
    y = bill_depth_mm,
    color = body_mass_g
  )) %>%
  v_scale_colour_gradient(
    name = "Body mass",
    low = "yellow",
    high = "red"
  )
```

---

temperatures *Temperature data*

Description

The dataset contains data about temperatures in France between 2018 and 2022.

Usage

```
temperatures
```

Format

A data frame with 365 observations and 6 variables.

Source

Enedis

top_cran_downloads *Top CRAN downloads*

Description

The dataset contains data about CRAN downloads retrieved with cranlogs::cran_top_downloads.

Usage

```
top_cran_downloads
```

Format

A data frame with 100 observations and 5 variables.

Source

cranlogs

top_generation *Top electricity-generating countries*

Description

This dataset represents the 10 countries with the highest electricity generation in 2023.

Usage

```
top_generation
```

Format

A data frame with 10 observations and 2 variables:

- `country` : Country name
- `electricity_generation` : Total electricity generation - Measured in terawatt-hours.

Source

Our World In Data

vchart *Create interactive charts with VChart*

Description

VChart is a charting component library, see more about it here : https://www.visactor.io/vchart.

Usage

```
vchart(
  data = NULL,
  mapping = NULL,
  ...,
  width = NULL,
  height = NULL,
  elementId = NULL
)
```

Arguments

data	Can be a data.frame if function used with other layers functions or a list of parameters for configuring a chart.
mapping	Default list of aesthetic mappings to use for chart, only used if data is a data.frame.
...	Additional parameters.
width	Fixed width for widget (in css units). The default is NULL, which results in intelligent automatic sizing based on the widget's container.
height	Fixed height for widget (in css units). The default is NULL, which results in intelligent automatic sizing based on the widget's container.
elementId	Use an explicit element ID for the widget (rather than an automatically generated one). Useful if you have other JavaScript that needs to explicitly discover and interact with a specific widget instance.

Value

A vchart() htmlwidget object.

Note

This function allow you to use JavaScript function VChart directly, see https://www.visactor.io/vchart/option/ for how to specify options.

Examples

```
library(vchartr)

# Use JS syntax to construct chart
vchart(
  type = "line",
  data = list(
    list(
      values = list(
        list(month = "January", value = 4.3),
        list(month = "February", value = 4.6),
        list(month = "March", value = 7.4),
        list(month = "April", value = 10.7),
        list(month = "May", value = 14.3),
        list(month = "June", value = 17.7),
        list(month = "July", value = 19.8),
        list(month = "August", value = 19.4),
        list(month = "September", value = 16.4),
        list(month = "October", value = 12.6),
        list(month = "November", value = 7.9),
        list(month = "December", value = 4.8)
      )
    )
  ),
  xField = "month",
  yField = "value",
  crosshair = list(
    xField = list(visible = TRUE)
  )
)

# or use R API
vchart(meteo_paris) %>%
  v_line(aes(month, temperature_avg)) %>%
  v_specs(
    crosshair = list(
      xField = list(visible = TRUE)
    )
  )

# or
vchart(meteo_paris, aes(month, temperature_avg)) %>%
  v_line() %>%
  v_specs(
    crosshair = list(
      xField = list(visible = TRUE)
    )
  )

# or
vchart() %>%
  v_line(aes(month, temperature_avg), data = meteo_paris) %>%
```

```
  v_specs(
    crosshair = list(
      xField = list(visible = TRUE)
    )
  )
```

---

vchart-shiny *Shiny bindings for vchart*

Description

Output and render functions for using vchart() within Shiny applications and interactive Rmd documents.

Usage

```
vchartOutput(outputId, width = "100%", height = "400px")

renderVchart(expr, env = parent.frame(), quoted = FALSE)
```

Arguments

outputId	output variable to read from
width, height	Must be a valid CSS unit (like "100%", "400px", "auto") or a number, which will be coerced to a string and have "px" appended.
expr	An expression that generates an HTML widget (or a promise of an HTML widget).
env	The environment in which to evaluate expr.
quoted	Is expr a quoted expression (with quote())? This is useful if you want to save an expression in a variable.

Value

An output or render function that enables the use of the widget within Shiny applications.

Examples

```
library(shiny)
library(bslib)
library(vchartr)

ui <- page_fluid(
  tags$div(
    style = "max-width: 900px; margin: auto;",
    tags$h2("vchart in shiny"),
    radioButtons(
      inputId = "data",
```

```
      label = "Show:",
      choiceNames = c("electricity mix by country", "countries generation by sources"),
      choiceValues = c("mix", "sources"),
      inline = TRUE
    ),
    conditionalPanel(
      condition = "input.data == 'mix'",
      selectInput(
        inputId = "country",
        label = "Select country:",
        choices = unique(electricity_mix$country)
      )
    ),
    conditionalPanel(
      condition = "input.data == 'sources'",
      selectInput(
        inputId = "source",
        label = "Select source:",
        choices = unique(electricity_mix$source)
      )
    ),
    vchartOutput(outputId = "mychart", height = "500px"),
    radioButtons(
      inputId = "type",
      label = "Represent as:",
      choices = c("bar", "pie", "treemap", "circlepacking"),
      inline = TRUE
    ),
    checkboxInput(
      inputId = "show_label",
      label = "Show label ?"
    )
  )
)

server <- function(input, output, session) {

  output$mychart <- renderVchart({

    if (input$data == "mix") {
      elec_data <- subset(electricity_mix, country == input$country)
      mapping <- aes(source, generation, fill = source)
    } else {
      elec_data <- subset(electricity_mix, source == input$source)
      mapping <- aes(country, generation, fill = country)
    }

    vc <- vchart(elec_data, mapping = mapping)
    if (input$type == "bar") {
      vc <- vc %>%
        v_bar(serie_id = "bar_serie") %>%
        v_specs_legend(visible = FALSE) %>%
        v_specs(
```

```
          xField = "x",
          label = list(visible = input$show_label),
          serie_id = "bar_serie"
        )
    } else if (input$type == "pie") {
      vc <- vc %>%
        v_pie(label = list(visible = input$show_label))
    } else if (input$type == "treemap") {
      vc <- vc %>%
        v_treemap(label = list(visible = input$show_label))
    } else if (input$type == "circlepacking") {
      vc <- vc %>%
        v_circlepacking(label = list(style = list(visible = input$show_label)))
    }
    vc %>%
      v_scale_color_manual(c(
        "oil" = "#80549f",
        "coal" = "#a68832",
        "solar" = "#d66b0d",
        "gas" = "#f20809",
        "wind" = "#72cbb7",
        "hydro" = "#2672b0",
        "nuclear" = "#e4a701"
      ))
  })

}

if (interactive())
  shinyApp(ui, server)
```

---

vmap *Create a Map*

Description

Create a Map

Usage

```
vmap(
  data,
  mapping = NULL,
  ...,
  projection = NULL,
  width = NULL,
  height = NULL,
  elementId = NULL
)
```

Arguments

data	An sf object.
mapping	efault list of aesthetic mappings to use for map.
...	Configuration options for the map.
projection	Geographical mapping type. See online documentation for the various possible choices.
width	Fixed width for widget (in css units). The default is NULL, which results in intelligent automatic sizing based on the widget's container.
height	Fixed height for widget (in css units). The default is NULL, which results in intelligent automatic sizing based on the widget's container.
elementId	Use an explicit element ID for the widget (rather than an automatically generated one). Useful if you have other JavaScript that needs to explicitly discover and interact with a specific widget instance.

Value

A vmap() htmlwidget object.

Examples

```
if (rlang::is_installed(c("sf", "geojsonio"))) {

  library(vchartr)
  library(sf)

  # Create map from sf object
  vmap(co2_world)

  # Draw data on the map
  vmap(co2_world, aes(name = name, fill = co2_per_capita))

  # Change projection and colors
  vmap(
    co2_world,
    aes(name = name, fill = co2_per_capita),
    projection = "equalEarth"
  ) %>%
    v_specs_colors(
      range = c(
        "#FFFFE5", "#FFF7BC", "#FEE391",
        "#FEC44F", "#FE9929", "#EC7014",
        "#CC4C02", "#993404", "#662506"
      )
    ) %>%
    v_specs_legend(
      orient = "bottom",
      type = "color",
      field = "fill"
    )
```

```
  # Map discrete data
  vmap(
    co2_world[!is.na(co2_world$co2_per_capita), ],
    aes(
      name = name,
      fill = ifelse(co2_per_capita >= 2.3, "Above", "Under")
    )
  ) %>%
    v_specs(
      area = list(
        style = list(
          stroke = "#FFFFFF"
        )
      )
    )

}
```

---

v_area *Create an Area Chart*

Description

Create an Area Chart

Usage

```
v_area(
  vc,
  mapping = NULL,
  data = NULL,
  name = NULL,
  stack = FALSE,
 area = list(style = list(curveType = "linear", fill = NULL, fillOpacity = NULL)),
  point = list(visible = FALSE),
  line = list(visible = FALSE),
  ...,
  serie_id = NULL,
  data_id = NULL
)
```

Arguments

vc	A chart initialized with vchart().
mapping	Default list of aesthetic mappings to use for chart.
data	Default dataset to use for chart. If not already a data.frame, it will be coerced to with as.data.frame.

name	Name for the serie, only used for single serie (no color/fill aesthetic supplied).
stack	Whether to stack the data or not (if fill aesthetic is provided).
area	Area's options, such as curve interpolation type, see online documentation.
point	Options for showing points on lines or not.
line	Options for showing lines or not.
...	Additional parameters for the serie.
data_id, serie_id	ID for the data/serie, can be used to further customize the chart with v_specs().

Value

A vchart() htmlwidget object.

Examples

```
library(vchartr)

# Basic Area Chart
vchart(eco2mix) %>%
  v_area(aes(date, solar))

# Two areas
vchart(eco2mix, aes(date)) %>%
  v_area(aes(y = wind)) %>%
  v_area(aes(y = solar))

# Line chart with discrete x axis
vchart(data.frame(month = month.abb, value = sample(1:50, 12))) %>%
  v_area(aes(month, value))

# Fill color
vchart(data.frame(month = month.abb, value = sample(1:50, 12))) %>%
  v_area(
    aes(month, value),
    area = list(
      style = list(fill = "firebrick", fill_opacity = 0.9)
    )
  )

# Smooth Area Chart
vchart(data.frame(month = month.abb, value = sample(1:50, 12))) %>%
  v_area(
    aes(month, value),
    area =  list(
      style = list(curveType = "monotone")
    )
  )

# Step Area Chart
```

```
vchart(data.frame(month = month.abb, value = sample(1:50, 12))) %>%
  v_area(
    aes(month, value),
    area = list(
      style = list(curveType = "stepAfter")
    )
  )

# Multiple areas
vchart(eco2mix_long) %>%
  v_area(aes(date, production, fill = source))

vchart(eco2mix_long) %>%
  v_area(
    aes(date, production, fill = source),
    stack = TRUE,
    area = list(
      style = list(fillOpacity = 1)
    )
  )


# Range area chart
vchart(temperatures, aes(date)) %>%
  v_area(aes(ymin = low, ymax = high)) %>%
  v_line(aes(y = average))

within(temperatures, {difference = `2024` - average}) %>%
  vchart(aes(date)) %>%
  v_area(
    aes(ymin = average, ymax = `2024`, difference = difference),
    area = list(
      style = list(
        fill = JS(
          "data => { return data.difference > 0 ? '#F68180' : '#2F64FF' ; }"
        ),
        fillOpacity = 1
      )
    )
  )
```

---

v_bar *Create a Bar Chart*

Description

Create a Bar Chart

Usage

```
v_bar(
  vc,
  mapping = NULL,
  data = NULL,
  name = NULL,
  stack = FALSE,
  percent = FALSE,
  direction = c("vertical", "horizontal"),
  ...,
  serie_id = NULL,
  data_id = NULL,
  data_specs = list()
)
```

Arguments

vc	A chart initialized with vchart().
mapping	Default list of aesthetic mappings to use for chart.
data	Default dataset to use for chart. If not already a data.frame, it will be coerced to with as.data.frame.
name	Name for the serie, only used for single serie (no color/fill aesthetic supplied).
stack	Whether to stack the data or not (if fill aesthetic is provided).
percent	Whether to display the data as a percentage.
direction	The direction configuration of the chart: "vertical" (default) or "horizontal".
...	Additional parameters for the serie.
data_id, serie_id	ID for the data/serie, can be used to further customize the chart with v_specs().
data_specs	Additional options for the data, see online documentation.

Value

A vchart() htmlwidget object.

Examples

```
library(vchartr)

# Classic Bar Chart
vchart(top_generation) %>%
  v_bar(aes(country, electricity_generation))

# Horizontal Bar Chart
vchart(top_generation) %>%
  v_bar(aes(country, electricity_generation), direction = "horizontal")
```

```
# Grouped Bar Chart
vchart(subset(world_electricity, type == "total")) %>%
  v_bar(aes(year, generation, fill = source))

# Horizontal Grouped Bar Chart
vchart(subset(world_electricity, type == "total")) %>%
  v_bar(aes(year, generation, fill = source), direction = "horizontal")

# Stacked Bar Chart
vchart(subset(world_electricity, type == "total")) %>%
  v_bar(aes(year, generation, fill = source), stack = TRUE)

# Percentage Stacked Bar Chart
vchart(subset(world_electricity, type == "total")) %>%
  v_bar(aes(year, generation, fill = source), stack = TRUE, percent = TRUE)
```

`v_boxplot` *Create a BoxPlot*

Description

Create a BoxPlot

Usage

```
v_boxplot(
  vc,
  mapping = NULL,
  data = NULL,
  name = NULL,
  ...,
  outliers = TRUE,
  args_outliers = NULL,
  serie_id = NULL,
  data_id = NULL
)
```

Arguments

`vc`	A chart initialized with `vchart()`.
`mapping`	Default list of aesthetic mappings to use for chart.
`data`	Default dataset to use for chart. If not already a `data.frame`, it will be coerced to with `as.data.frame`.
`name`	Name for the serie, only used for single serie (no `color`/`fill` aesthetic supplied).
`...`	Arguments passed to JavaScript methods.

outliers Display or not outliers.

args_outliers Arguments passed to v_scatter().

data_id, serie_id
ID for the data/serie, can be used to further customize the chart with v_specs().

Value

A vchart() htmlwidget object.

Examples

```
library(vchartr)

data("penguins", package = "palmerpenguins")

vchart(penguins) %>%
  v_boxplot(aes(species, flipper_length_mm))

vchart(penguins) %>%
  v_boxplot(aes(species, flipper_length_mm, color = sex))


data("mpg", package = "ggplot2")

vchart(mpg) %>%
  v_boxplot(aes(as.character(year), hwy))

vchart(mpg) %>%
  v_boxplot(aes(class, hwy))

vchart(mpg) %>%
  v_boxplot(aes(class, hwy, color = as.character(year)))
```

---

v_circlepacking *Create a Circle Packing Chart*

Description

Create a Circle Packing Chart

Usage

```
v_circlepacking(
  vc,
  mapping = NULL,
  data = NULL,
  name = NULL,
  drill = TRUE,
  use_root = FALSE,
```

```
  fill_opacity = JS("d => d.isLeaf ? 0.75 : 0.25;"),
  label_visible = JS("d => d.depth === 1;"),
  ...,
  serie_id = NULL,
  data_id = NULL
)
```

Arguments

vc	A chart initialized with vchart().
mapping	Default list of aesthetic mappings to use for chart.
data	Default dataset to use for chart. If not already a data.frame, it will be coerced to with as.data.frame.
name	Name for the serie, only used for single serie (no color/fill aesthetic supplied).
drill	Drill-down function switch.
use_root	Add a root level in the hierarchy, can be TRUE (in this case root level will be named root) or a character (use as the name for the root level).
fill_opacity	Fill opacity, a JS function determining the opacity of the elements.
label_visible	A JS function to control visibility of labels.
...	Additional parameters for the serie.
data_id, serie_id	ID for the data/serie, can be used to further customize the chart with v_specs().

Value

A vchart() htmlwidget object.

Examples

```
library(vchartr)

vchart(countries_gdp) %>%
  v_circlepacking(
    aes(lvl1 = REGION_UN, lvl2 = SUBREGION, lvl3 = ADMIN, value = GDP_MD)
  )


# With root level
vchart(countries_gdp) %>%
  v_circlepacking(
    aes(lvl1 = REGION_UN, lvl2 = SUBREGION, lvl3 = ADMIN, value = GDP_MD),
    use_root = "World"
  )


# Custom colors
vchart(countries_gdp) %>%
```

```
  v_circlepacking(
    aes(lvl1 = REGION_UN, lvl2 = SUBREGION, lvl3 = ADMIN, value = GDP_MD)
  ) %>%
  v_scale_color_manual(c(
    Oceania = "#E6AB02",
    Africa = "#1B9E77",
    Americas = "#D95F02",
    Asia = "#E7298A",
    Europe = "#66A61E",
    Antarctica = "#7570B3"
  ))

# Bubble Chart
vchart(countries_gdp) %>%
  v_circlepacking(
    aes(ADMIN, GDP_MD),
    label_visible = JS("d => d.value > 261921;"), # 261921 = 3rd Qu.
  )
```

`v_event` *VChart events*

Description

VChart events

Usage

```
v_event(vc, name, params, fun, ...)
```

Arguments

`vc`	A chart initialized with `vchart()`.
`name`	Name of the event, e.g. `"click"`.
`params`	Parameters to specifically monitor events in a certain part of the chart.
`fun`	JavaScript function executed when the event occurs.
`...`	Not used.

Value

A `vchart()` `htmlwidget` object.

Examples

```
library(vchartr)

vchart(top_generation) %>%
  v_bar(aes(country, electricity_generation)) %>%
  v_event(
    name = "click",
    params = list(level = "mark", type = "bar"),
    fun = JS(
      "e => {",
      " console.log(e);",
      " alert(e.datum.x);",
      "}"
    )
  )
```

`v_facet_wrap` *Facets for vchart*

Description

Create matrix of charts by row and column faceting variable (`v_facet_grid`), or by specified number of row and column for faceting variable(s) (`v_facet_wrap`).

Usage

```
v_facet_wrap(
  vc,
  facets,
  nrow = NULL,
  ncol = NULL,
  scales = c("fixed", "free", "free_y", "free_x"),
  labeller = label_value
)
```

Arguments

`vc`	A chart initialized with `vchart()`.
`facets`	Variable(s) to use for facetting, wrapped in `vars(...)`.
`nrow, ncol`	Number of row and column in output matrix.
`scales`	Should scales be fixed (`"fixed"`, the default), free (`"free"`), or free in one dimension (`"free_x"`, `"free_y"`)?
`labeller`	A function with one argument containing for each facet the value of the faceting variable.

Value

A vchart() htmlwidget object.

Examples

```
library(vchartr)
library(ggplot2)

# Use vars() to supply faceting variables:
vchart(mpg) %>%
  v_scatter(aes(displ, hwy)) %>%
  v_facet_wrap(vars(class))

# Control the number of rows and columns with nrow and ncol
vchart(mpg) %>%
  v_scatter(aes(displ, hwy)) %>%
  v_facet_wrap(vars(class), ncol = 3)

# You can facet by multiple variables
vchart(mpg) %>%
  v_scatter(aes(displ, hwy)) %>%
  v_facet_wrap(vars(cyl, drv))

# Use the `labeller` option to control how labels are printed:
vchart(mpg) %>%
  v_scatter(aes(displ, hwy)) %>%
  v_facet_wrap(vars(cyl, drv), labeller = label_both)

# To change the order in which the panels appear, change the levels
# of the underlying factor.
mpg$class2 <- reorder(mpg$class, mpg$displ)
vchart(mpg) %>%
  v_scatter(aes(displ, hwy)) %>%
  v_facet_wrap(vars(class2), ncol = 3)

# By default, the same scales are used for all panels. You can allow
# scales to vary across the panels with the `scales` argument.
vchart(mpg) %>%
  v_scatter(aes(displ, hwy)) %>%
  v_facet_wrap(vars(class), scales = "free")
```

---

v_gauge *Create a Gauge Chart*

Description

Create a Gauge Chart

Usage

```
v_gauge(
  vc,
  mapping = NULL,
  data = NULL,
  name = NULL,
  outerRadius = 0.8,
  innerRadius = 0.75,
  startAngle = -240,
  endAngle = 60,
  ...,
  serie_id = NULL,
  data_id = NULL
)
```

Arguments

vc	A chart initialized with vchart().
mapping	Default list of aesthetic mappings to use for chart.
data	Default dataset to use for chart. If not already a data.frame, it will be coerced to with as.data.frame.
name	Name for the serie, only used for single serie (no color/fill aesthetic supplied).
outerRadius	Sector outer radius, with a numerical range of 0 - 1.
innerRadius	Sector inner radius, with a numerical range of 0 - 1.
startAngle	Starting angle of the sector. In degrees.
endAngle	Ending angle of the sector. In degrees.
...	Additional parameters for the serie.
data_id, serie_id	ID for the data/serie, can be used to further customize the chart with v_specs().

Value

A vchart() htmlwidget object.

Examples

```
library(vchartr)

vchart() %>%
  v_gauge(aes("My gauge", 0.8))

vchart() %>%
  v_gauge(
    aes("My gauge", 0.8),
    gauge = list(
      type = "circularProgress",
```

```
      cornerRadius = 20,
      progress = list(
        style = list(
          fill = "forestgreen"
        )
      ),
      track = list(
        style = list(
          fill = "#BCBDBC"
        )
      )
    ),
    pointer = list(
      style = list(
        fill = "#2F2E2F"
      )
    )
  )


vchart() %>%
  v_gauge(aes("My gauge", 0.8)) %>%
  v_scale_y_continuous(labels = ".0%")
```

---

v_heatmap *Create a Heatmap Chart*

Description

Create a Heatmap Chart

Usage

```
v_heatmap(
  vc,
  mapping = NULL,
  data = NULL,
  name = NULL,
  ...,
  serie_id = NULL,
  data_id = NULL
)
```

Arguments

vc	A chart initialized with vchart().
mapping	Default list of aesthetic mappings to use for chart.
data	Default dataset to use for chart. If not already a data.frame, it will be coerced to with as.data.frame.

name — Name for the serie, only used for single serie (no color/fill aesthetic supplied).

... — Additional parameters for the serie.

data_id, serie_id — ID for the data/serie, can be used to further customize the chart with v_specs().

Value

A vchart() htmlwidget object.

Examples

```
library(vchartr)

# Heatmap with continuous fill variable
vchart(co2_emissions) %>%
  v_heatmap(aes(x = year, y = country, fill = co2_per_capita))

# Change colors
vchart(co2_emissions) %>%
  v_heatmap(aes(x = year, y = country, fill = co2_per_capita)) %>%
  v_specs_colors(
    range = rev(
      c("#8C510A", "#BF812D", "#DFC27D", "#F6E8C3",
        "#C7EAE5", "#80CDC1", "#35978F", "#01665E")
    )
  )

# Heatmap with discrete fill variable
vchart(co2_emissions) %>%
  v_heatmap(aes(x = year, y = country, fill = co2_growth_change))

# Change colors
vchart(co2_emissions) %>%
  v_heatmap(aes(x = year, y = country, fill = co2_growth_change)) %>%
  v_scale_fill_manual(c(
    Increase = "firebrick",
    Decrease = "forestgreen"
  ))
```

---

v_hist *Create an Histogram*

Description

Create an Histogram

Usage

```
v_hist(
  vc,
  mapping = NULL,
  data = NULL,
  name = NULL,
  stack = FALSE,
  bins = 30,
  binwidth = NULL,
  ...,
  serie_id = NULL,
  data_id = NULL
)
```

Arguments

vc	A chart initialized with vchart().
mapping	Default list of aesthetic mappings to use for chart.
data	Default dataset to use for chart. If not already a data.frame, it will be coerced to with as.data.frame.
name	Name for the serie, only used for single serie (no color/fill aesthetic supplied).
stack	Whether to stack the data or not (if fill aesthetic is provided).
bins	Number of bins. Overridden by binwidth. Defaults to 30.
binwidth	The width of the bins. Can be specified as a numeric value or as a function that takes x after scale transformation as input and returns a single numeric value. When specifying a function along with a grouping structure, the function will be called once per group. The default is to use the number of bins in bins, covering the range of the data. You should always override this value, exploring multiple widths to find the best to illustrate the stories in your data. The bin width of a date variable is the number of days in each time; the bin width of a time variable is the number of seconds.
...	Additional properties for histogram bars.
data_id, serie_id	ID for the data/serie, can be used to further customize the chart with v_specs().

Value

A vchart() htmlwidget object.

Examples

```
library(vchartr)
library(palmerpenguins)

# Create an histogram using a numeric variable
```

```
vchart(penguins) %>%
  v_hist(aes(flipper_length_mm))

# Customize some style properties
vchart(penguins) %>%
  v_hist(
    aes(flipper_length_mm),
    bar = list(
      style = list(
        stroke = "white",
        line_width = 1,
        fill = "forestgreen"
      )
    )
  )

# Use fill aesthetic to differentiate series
vchart(penguins) %>%
  v_hist(aes(flipper_length_mm, fill = species))

# Stack results
vchart(penguins) %>%
  v_hist(aes(flipper_length_mm, fill = species), stack = TRUE)

# Use custom colors
vchart(penguins) %>%
  v_hist(
    aes(flipper_length_mm, fill = species),
    bar = list(
      style = list(opacity = 0.5)
    )
  ) %>%
  v_scale_color_manual(c(
    Adelie = "#ffa232",
    Chinstrap = "#33a2a2",
    Gentoo = "#b34df2"
  ))
```

`v_jitter` *Create Jittered Points Scatter Chart*

Description

Create Jittered Points Scatter Chart

Usage

```
v_jitter(
  vc,
```

```
  mapping = NULL,
  data = NULL,
  name = NULL,
  width = NULL,
  height = NULL,
  ...,
  serie_id = NULL,
  data_id = NULL
)
```

Arguments

vc	A chart initialized with vchart().
mapping	Default list of aesthetic mappings to use for chart.
data	Default dataset to use for chart. If not already a data.frame, it will be coerced to with as.data.frame.
name	Name for the serie, only used for single serie (no color/fill aesthetic supplied).
width, height	Amount of vertical and horizontal jitter. The jitter is added in both positive and negative directions, so the total spread is twice the value specified here. If omitted, defaults to 40% of the resolution of the data: this means the jitter values will occupy 80% of the implied bins. Categorical data is aligned on the integers, so a width or height of 0.5 will spread the data so it's not possible to see the distinction between the categories.
...	Additional parameters for the serie.
data_id, serie_id	ID for the data/serie, can be used to further customize the chart with v_specs().

Value

A vchart() htmlwidget object.

Examples

```
library(vchartr)

data("mpg", package = "ggplot2")
data("penguins", package = "palmerpenguins")

# With continuous x
vchart(mpg) %>%
  v_jitter(aes(cyl, hwy))

# with discrete x
vchart(penguins) %>%
  v_jitter(aes(species, bill_length_mm))

# Colour points
vchart(mpg) %>%
```

```
  v_jitter(aes(cyl, hwy, colour = class))

# Use smaller width/height to emphasise categories
vchart(mpg) %>%
  v_jitter(aes(cyl, hwy), width = 0.25)

# Use larger width/height to completely smooth away discreteness
vchart(mpg) %>%
  v_jitter(aes(cty, hwy), width = 0.5, height = 0.5)
```

---

v_labs *Set chart title and subtitle*

Description

Set chart title and subtitle

Usage

```
v_labs(vc, title = NULL, subtitle = NULL, x = NULL, y = NULL)
```

Arguments

vc	An htmlwidget created with vchart().
title	Title for the chart.
subtitle	Subtitle for the chart.
x, y	Axes titles.

Value

A vchart() htmlwidget object.

Examples

```
library(vchartr)
data("mpg", package = "ggplot2")

vchart(table(Class = mpg$class), aes(Class, Freq)) %>%
  v_bar() %>%
  v_labs(
    title = "Title for the chart",
    subtitle = "A subtitle to be placed under the title"
  )
```

v_line *Create a Line Chart*

Description

Create a Line Chart

Usage

```
v_line(
  vc,
  mapping = NULL,
  data = NULL,
  name = NULL,
  line = list(style = list(curveType = "linear", lineDash = 0, stroke = NULL)),
  point = list(visible = FALSE),
  ...,
  serie_id = NULL,
  data_id = NULL
)
```

Arguments

vc	A chart initialized with vchart().
mapping	Default list of aesthetic mappings to use for chart.
data	Default dataset to use for chart. If not already a data.frame, it will be coerced to with as.data.frame.
name	Name for the serie, only used for single serie (no color/fill aesthetic supplied).
line	Line's options, such as curve interpolation type, see online documentation
point	Options for showing points on lines or not.
...	Additional parameters for the serie.
data_id, serie_id	ID for the data/serie, can be used to further customize the chart with v_specs().

Value

A vchart() htmlwidget object.

Examples

```
library(vchartr)

# Basic Line Chart
vchart(eco2mix) %>%
  v_line(aes(date, solar))
```

```
# Two lines
vchart(tail(eco2mix, 30), aes(date)) %>%
  v_line(aes(y = solar)) %>%
  v_line(aes(y = wind))

# Line chart with discrete x axis
vchart(data.frame(month = month.abb, value = sample(1:50, 12))) %>%
  v_line(aes(month, value))

# Stroke color
vchart(data.frame(month = month.abb, value = sample(1:50, 12))) %>%
  v_line(
    aes(month, value),
    line = list(style = list(stroke = "firebrick"))
  )

# Smooth Line Chart
vchart(data.frame(month = month.abb, value = sample(1:50, 12))) %>%
  v_line(
    aes(month, value),
    line = list(style = list(curveType = "monotone"))
  )

# Step Line Chart
vchart(data.frame(month = month.abb, value = sample(1:50, 12))) %>%
  v_line(
    aes(month, value),
    line = list(style = list(curveType = "stepAfter"))
  )

# Dash array
vchart(data.frame(month = month.abb, value = sample(1:50, 12))) %>%
  v_line(
    aes(month, value),
    line = list(style = list(lineDash = c(10, 10)))
  )

# Multiple lines
vchart(eco2mix_long) %>%
  v_line(aes(date, production, color = source))
```

`v_pie` *Create a Pie Chart*

Description

Create a Pie Chart

Usage

```
v_pie(
  vc,
  mapping = NULL,
  data = NULL,
  name = NULL,
  label = list(visible = TRUE),
  ...,
  serie_id = NULL,
  data_id = NULL
)
```

Arguments

vc	A chart initialized with vchart().
mapping	Default list of aesthetic mappings to use for chart.
data	Default dataset to use for chart. If not already a data.frame, it will be coerced to with as.data.frame.
name	Name for the serie, only used for single serie (no color/fill aesthetic supplied).
label	Options for displaying labels on the pie chart.
...	Additional parameters for the serie.
data_id, serie_id	ID for the data/serie, can be used to further customize the chart with v_specs().

Value

A vchart() htmlwidget object.

Examples

```
library(vchartr)

# Basic Pie Chart
subset(world_electricity, year == 2023 & type == "total") %>%
  vchart() %>%
  v_pie(aes(x = source, y = generation))

# Use custom colors
subset(world_electricity, year == 2023 & type == "total") %>%
  vchart() %>%
  v_pie(aes(x = source, y = generation)) %>%
  v_scale_color_manual(c(
    "Low carbon" = "#a3be8c",
    "Fossil fuels" = "#4C566A"
  ))

# Customize tooltip
```

```
subset(world_electricity, year == 2023 & type == "total") %>%
  vchart() %>%
  v_pie(aes(x = source, y = generation)) %>%
  v_specs_tooltip(
    mark = list(
      content = list(
        list(
          key = JS("datum => datum['x']"),
          value = JS("datum => Math.round(datum['y']) + ' TWh'")
        ),
        list(
          hasShape = FALSE,
          key = "Proportion",
          value = JS("datum => datum._percent_ + '%'")
        )
      )
    )
  )


# Nested Pie Chart
vchart() %>%
  v_pie(
    data = subset(world_electricity, year == 2023 & type == "total"),
    mapping = aes(x = source, y = generation),
    outerRadius = 0.65,
    innerRadius = 0,
    label = list(
      visible = TRUE,
      position = "inside",
      rotate = FALSE,
      style = list(fill = "white")
    ),
    pie = list(
      style = list(
        stroke = "#FFFFFF",
        lineWidth = 2
      )
    )
  ) %>%
  v_pie(
    data = subset(world_electricity, year == 2023 & type == "detail"),
    mapping = aes(x = source, y = generation),
    outerRadius = 0.8,
    innerRadius = 0.67,
    pie = list(
      style = list(
        stroke = "#FFFFFF",
        lineWidth = 2
      )
    )
  )
```

v_progress *Create a Progress Chart*

Description

Create a Progress Chart

Usage

```
v_progress(
  vc,
  mapping = NULL,
  data = NULL,
  name = NULL,
  ...,
  serie_id = NULL,
  data_id = NULL
)
```

Arguments

vc	A chart initialized with vchart().
mapping	Default list of aesthetic mappings to use for chart.
data	Default dataset to use for chart. If not already a data.frame, it will be coerced to with as.data.frame.
name	Name for the serie, only used for single serie (no color/fill aesthetic supplied).
...	Additional parameters for the serie.
data_id, serie_id	ID for the data/serie, can be used to further customize the chart with v_specs().

Value

A vchart() htmlwidget object.

Examples

```
library(vchartr)

vchart() %>%
  v_progress(aes(0.85, "My progress"))

data.frame(
  x = c(0.4, 0.3, 0.8, 0.6),
  y = paste("Course", 1:4)
) %>%
  vchart() %>%
```

```
  v_progress(
    aes(x, y),
    cornerRadius = 20,
    bandWidth = 30
  ) %>%
  v_scale_y_discrete(
    label = list(visible = TRUE),
    domainLine = list(visible = FALSE)
  )
```

---

v_radar *Create a Radar Chart*

Description

Create a Radar Chart

Usage

```
v_radar(
  vc,
  mapping = NULL,
  data = NULL,
  name = NULL,
  ...,
  serie_id = NULL,
  data_id = NULL
)
```

Arguments

vc	A chart initialized with vchart().
mapping	Default list of aesthetic mappings to use for chart.
data	Default dataset to use for chart. If not already a data.frame, it will be coerced to with as.data.frame.
name	Name for the serie, only used for single serie (no color/fill aesthetic supplied).
...	Additional parameters for the serie.
data_id, serie_id	ID for the data/serie, can be used to further customize the chart with v_specs().

Value

A vchart() htmlwidget object.

Examples

```
library(vchartr)

# Default radar chart
subset(electricity_mix, country == "Germany") %>%
  vchart() %>%
  v_radar(aes(source, generation))

# Without area
subset(electricity_mix, country == "Germany") %>%
  vchart() %>%
  v_radar(
    aes(source, generation),
    area = list(visible = FALSE)
  )


# Mutliple series
subset(electricity_mix, country %in% c("Germany", "Canada")) %>%
  vchart() %>%
  v_radar(aes(source, generation, color = country))



# Custom axes
subset(electricity_mix, country == "Germany") %>%
  vchart() %>%
  v_radar(aes(source, generation)) %>%
  v_scale_y_continuous(min = 0, max = 200)

subset(electricity_mix, country == "Germany") %>%
  vchart() %>%
  v_radar(aes(source, generation)) %>%
  v_scale_y_continuous(
    grid = list(smooth = FALSE),
    domainLine = list(visible = FALSE)
  ) %>%
  v_scale_x_discrete(
    label = list(space = 20),
    domainLine = list(visible = FALSE)
  )
```

---

v_sankey *Create a Sankey Chart*

Description

Create a Sankey Chart

Usage

```
v_sankey(
  vc,
  mapping = NULL,
  data = NULL,
  name = NULL,
  ...,
  serie_id = NULL,
  data_id = NULL
)
```

Arguments

vc	A chart initialized with vchart().
mapping	Default list of aesthetic mappings to use for chart.
data	Default dataset to use for chart. If not already a data.frame, it will be coerced to with as.data.frame.
name	Name for the serie, only used for single serie (no color/fill aesthetic supplied).
...	Additional parameters for the serie.
data_id, serie_id	ID for the data/serie, can be used to further customize the chart with v_specs().

Value

A vchart() htmlwidget object.

Examples

```
library(vchartr)

# Basic Sankey Chart
vchart(energy_sankey) %>%
  v_sankey(aes(target, source, value = value))

# Some options
vchart(energy_sankey) %>%
  v_sankey(
    aes(target, source, value = value),
    nodeAlign = "left",
    nodeGap = 8,
    nodeWidth = 10,
    minNodeHeight = 4,
    link = list(
      state = list(
        hover = list(
          fillOpacity = 1
        )
      )
```

```
    )
  )


# With data as tree structure
titanic <- as.data.frame(Titanic)
vchart(titanic) %>%
  v_sankey(aes(
    lvl1 = Class,
    lvl2 = Sex,
    lvl3 = Age,
    lvl4 = Survived,
    value = Freq
  ))


# Only one level
titanic_class <- titanic %>%
  aggregate(data = ., Freq ~ Class + Survived, FUN = sum)

vchart(titanic_class) %>%
  v_sankey(aes(Survived, Class, value = Freq))
```

`v_scale_size` *Size scale for continuous data*

Description

Size scale for continuous data

Usage

```
v_scale_size(
  vc,
  name = NULL,
  range = c(5, 30),
  ...,
  position = c("right", "bottom", "left", "top"),
  align = c("middle", "start", "end")
)
```

Arguments

`vc`	An htmlwidget created with `vchart()` or specific chart's type function.
`name`	Title for the legend.
`range`	Range of sizes for the points plotted.

...	Additional parameters for the legend.
position	Position of the legend.
align	Alignment of the legend.

Value

A vchart() htmlwidget object.

Examples

```
library(vchartr)
data("penguins", package = "palmerpenguins")

vchart(penguins) %>%
  v_scatter(aes(
    x = bill_length_mm,
    y = bill_depth_mm,
    size = body_mass_g
  )) %>%
  v_scale_size(
    name = "Body mass",
    range = c(1, 20)
  )
```

---

v_scatter *Create a Scatter Chart*

Description

Create a Scatter Chart

Usage

```
v_scatter(
  vc,
  mapping = NULL,
  data = NULL,
  name = NULL,
  ...,
  serie_id = NULL,
  data_id = NULL
)
```

Arguments

vc	A chart initialized with vchart().
mapping	Default list of aesthetic mappings to use for chart.
data	Default dataset to use for chart. If not already a data.frame, it will be coerced to with as.data.frame.
name	Name for the serie, only used for single serie (no color/fill aesthetic supplied).
...	Additional parameters for the serie.
data_id, serie_id	ID for the data/serie, can be used to further customize the chart with v_specs().

Value

A vchart() htmlwidget object.

Examples

```
library(vchartr)
data("penguins", package = "palmerpenguins")

# Basic scatter chart
vchart(penguins) %>%
  v_scatter(aes(x = flipper_length_mm, y = body_mass_g))

# Color series with discrete values
vchart(penguins) %>%
  v_scatter(aes(x = flipper_length_mm, y = body_mass_g, color = species))

# Color series with continuous values
vchart(penguins) %>%
  v_scatter(aes(x = bill_length_mm, y = bill_depth_mm, color = body_mass_g))

# Size of points
vchart(penguins) %>%
  v_scatter(aes(x = bill_length_mm, y = bill_depth_mm, size = body_mass_g))

# Size and color
vchart(penguins) %>%
  v_scatter(aes(
    x = bill_length_mm,
    y = bill_depth_mm,
    color = body_mass_g,
    size = body_mass_g
  ))

# With shapes
vchart(penguins) %>%
  v_scatter(
    aes(
      x = bill_length_mm,
```

```
      y = bill_depth_mm,
      color = species,
      shape = species
    )
  )



vchart(penguins) %>%
  v_scatter(
    aes(x = flipper_length_mm, y = body_mass_g, color = species)
  ) %>%
  v_scale_color_manual(c(
    Adelie = "#ffa232",
    Chinstrap = "#33a2a2",
    Gentoo = "#b34df2"
  ))
```

---

v_smooth *Create an Smooth Line Chart*

Description

Create an Smooth Line Chart

Usage

```
v_smooth(
  vc,
  mapping = NULL,
  data = NULL,
  name = NULL,
  method = NULL,
  formula = NULL,
  se = TRUE,
  n = 80,
  span = 0.75,
  ...,
  args_area = NULL,
  serie_id = NULL,
  data_id = NULL
)
```

Arguments

vc	A chart initialized with vchart().
mapping	Default list of aesthetic mappings to use for chart.

data	Default dataset to use for chart. If not already a data.frame, it will be coerced to with as.data.frame.
name	Name for the serie, only used for single serie (no color/fill aesthetic supplied).
method	Smoothing method (function) to use, accepts either NULL or a character vector, e.g. "lm", "glm", "gam", "loess" or a function, e.g. MASS::rlm or mgcv::gam, stats::lm, or stats::loess. "auto" is also accepted for backwards compatibility. It is equivalent to NULL. For method = NULL the smoothing method is chosen based on the size of the largest group (across all panels). stats::loess() is used for less than 1,000 observations; otherwise mgcv::gam() is used with formula = y ~ s(x, bs = "cs") with method = "REML". Somewhat anecdotally, loess gives a better appearance, but is $O(N^2)$ in memory, so does not work for larger datasets. If you have fewer than 1,000 observations but want to use the same gam() model that method = NULL would use, then set method = "gam", formula = y ~ s(x, bs = "cs").
formula	Formula to use in smoothing function, eg. y ~ x, y ~ poly(x, 2), y ~ log(x). NULL by default, in which case method = NULL implies formula = y ~ x when there are fewer than 1,000 observations and formula = y ~ s(x, bs = "cs") otherwise.
se	Display confidence band around smooth? (TRUE by default, see level to control.)
n	Number of points at which to evaluate smoother.
span	Controls the amount of smoothing for the default loess smoother. Smaller numbers produce wigglier lines, larger numbers produce smoother lines. Only used with loess, i.e. when method = "loess", or when method = NULL (the default) and there are fewer than 1,000 observations.
...	Additional parameters for lines.
args_area	Arguments for area.
data_id, serie_id	ID for the data/serie, can be used to further customize the chart with v_specs().

Value

A vchart() htmlwidget object.

Examples

```
library(vchartr)

data("mpg", package =  "ggplot2")

vchart(mpg, aes(displ, hwy)) %>%
  v_smooth()

vchart(mpg, aes(displ, hwy)) %>%
  v_smooth(se = FALSE)
```

```
vchart(mpg, aes(displ, hwy, color = class)) %>%
  v_smooth()
```

---

v_specs *Specify configuration options for a* vchart().

Description

Specify configuration options for a vchart().

Usage

```
v_specs(vc, ..., serie_id = NULL, drop_nulls = FALSE)
```

Arguments

vc	An htmlwidget created with vchart().
...	List of options to specify for the chart, see https://www.visactor.io/vchart/option/.
serie_id	Used to set or modify options for a chart where there are multiple series. You can use : • a numeric to target the position of the serie in the order where it's added to the chart • a character to refer to a serie_id set when the serie was added to the plot.
drop_nulls	Drop NULL elements from the options.

Value

A vchart() htmlwidget object.

Examples

```
library(vchartr)
data("mpg", package = "ggplot2")

vchart(table(Class = mpg$class)) %>%
  v_bar(aes(Class, Freq)) %>%
  v_specs(
    label = list(visible = TRUE),
    color = list("firebrick")
  )
```

v_specs_axes *Axes configuration*

Description

Axes configuration

Usage

```
v_specs_axes(
  vc,
  position = c("left", "bottom", "right", "top", "angle", "radius"),
  ...,
  remove = FALSE
)
```

Arguments

vc	An htmlwidget created with vchart().
position	Position of the axe on the chart.
...	Configuration options.
remove	If TRUE then axe is removed and other parameters are ignored.

Value

A vchart() htmlwidget object.

Examples

```
library(vchartr)

# Configure some options for axes
vchart() %>%
  v_line(aes(x = month.name, y = sample(5:25, 12))) %>%
  v_specs_axes(
    position = "left",
    title = list(
      visible = TRUE,
      text = "Y axis title",
      position = "start"
    ),
    label = list(
      formatMethod = JS("val => `${val}°C`")
    ),
    domainLine = list(
      visible = TRUE,
      style = list(stroke = "#000")
    ),
```

```
    tick = list(
      visible = TRUE,
      tickStep = 2,
      tickSize = 6,
      style = list(stroke = "#000")
    ),
    grid = list(
      visible = TRUE,
      style = list(lineDash = list(0), stroke = "#6E6E6E", zIndex = 100)
    )
  )%>%
  v_specs_axes(
    position = "bottom",
    title = list(
      visible = TRUE,
      text = "X axis title",
      position = "end"
    ),
    domainLine = list(
      visible = TRUE,
      style = list(stroke = "#000")
    ),
    tick = list(
      visible = TRUE,
      tickStep = 2,
      tickSize = 6,
      style = list(stroke = "#000")
    ),
    grid = list(
      visible = TRUE,
      style = list(lineDash = list(0)),
      alternateColor = c("#F2F2F2", "#FFFFFF"),
      alignWithLabel = TRUE
    )
  )



# By default vline add an axe on the left
vchart() %>%
  v_line(aes(x = month.name, y = sample(5:25, 12))) %>%
  v_specs_axes(position = "left", remove = TRUE) %>%
  v_specs_axes(position = "right", type = "linear")

# Use secondary axes
vchart() %>%
  v_line(aes(x = month.name, y = sample(5:25, 12)), serie_id = "serie_left") %>%
  v_line(aes(x = month.name, y = sample(5:25 * 100, 12)), serie_id = "serie_right") %>%
  v_specs_axes(position = "left", seriesId = "serie_left") %>%
  v_specs_axes(position = "right", type = "linear", seriesId = "serie_right")
```

v_specs_colors *Set color(s) for chart*

Description

Set color(s) for chart

Usage

```
v_specs_colors(vc, ...)
```

Arguments

vc	An htmlwidget created with vchart().
...	Colors options, can be a single color code, a vector of colors to use or a list with more options. For v_colors_manual it should be a named list with data values as name and color as values.

Value

A vchart() htmlwidget object.

Examples

```
library(vchartr)
data("mpg", package = "ggplot2")

vchart(table(Class = mpg$class)) %>%
  v_bar(aes(Class, Freq)) %>%
  v_specs_colors("#8FBCBB")
```

v_specs_crosshair *Add crosshair to chart*

Description

Add crosshair to chart

Usage

```
v_specs_crosshair(vc, ...)
```

Arguments

vc	An htmlwidget created with vchart().
...	Options for the legend, see examples or online documentation.

Value

A vchart() htmlwidget object.

Examples

```
library(vchartr)

data.frame(month = month.abb, value = sample(1:50, 12)) %>%
  vchart() %>%
  v_line(aes(month, value)) %>%
  v_specs_crosshair(
    xField = list(
      visible = TRUE,
      line = list(type = "rect"),
      defaultSelect = list(
        axisIndex = 0,
        datum = "May"
      ),
      label = list(visible = TRUE)
    ),
    yField = list(
      visible = TRUE,
      defaultSelect = list(
        axisIndex = 1,
        datum = 30
      ),
      line = list(
        style = list(
          lineWidth = 1,
          opacity = 1,
          stroke = "#000",
          lineDash = c(2, 2)
        )
      ),
      label = list(visible = TRUE)
    )
  )
```

---

v_specs_custom_mark *Add custom mark to chart*

Description

Add custom mark to chart

Usage

```
v_specs_custom_mark(vc, ...)
```

Arguments

vc	An htmlwidget created with vchart().
...	Options for the legend, see examples or online documentation.

Value

A vchart() htmlwidget object.

Examples

```
library(vchartr)

world_electricity %>%
  subset(type == "detail") %>%
  vchart() %>%
  v_bar(
    aes(source, generation, player = year),
    direction = "h",
    data_id = "mydata"
  ) %>%
  v_specs_custom_mark(
    type = "text",
    dataId = "mydata",
    style = list(
      textBaseline = "bottom",
      fontSize = 60,
      textAlign = "right",
      fontWeight = 700,
      text = JS("datum => datum.player"),
      x = JS(
        "(datum, ctx) => {",
        " return ctx.vchart.getChart().getCanvasRect().width - 50;",
        "}"
      ),
      y = JS(
        "(datum, ctx) => {",
        " return ctx.vchart.getChart().getCanvasRect().height - 150;",
        "}"
      ),
      fill = "grey",
      fillOpacity = 0.5
    )
  )
```

---

v_specs_datazoom *Add data zoom to a chart*

Description

Add data zoom to a chart

Usage

```
v_specs_datazoom(
  vc,
  start = "{label:%Y-%m-%d}",
  end = "{label:%Y-%m-%d}",
  ...,
  brush = TRUE
)
```

Arguments

vc	A chart created with vchart().
start, end	Formatter for the start/end label, e.g. : "Start: \{label:%Y-%m-%d\}", where the part between braces will be replaced by the date with the format specified.
...	Additional parameters for dataZoom property, see online documentation.
brush	Logical, add the ability to brush the chart to zoom in.

Value

A vchart() htmlwidget object.

Examples

```
library(vchartr)

data("economics", package = "ggplot2")
vchart(economics, aes(date, unemploy)) %>%
  v_line() %>%
  v_specs_datazoom()


co2_emissions %>%
  subset(country %in% c("China", "United States", "India")) %>%
  vchart() %>%
  v_line(aes(year, co2, color = country)) %>%
  v_specs_datazoom(start = "{label:.0f}", startValue = 1990, end = "{label:.0f}")
```

---

v_specs_indicator *Add indicator to chart*

Description

Add indicator to chart

Usage

```
v_specs_indicator(vc, ...)
```

Arguments

vc	An htmlwidget created with vchart().
...	Options for the legend, see examples or online documentation.

Value

A vchart() htmlwidget object.

Examples

```
library(vchartr)

electricity_mix %>%
  subset(country == "France") %>%
  vchart() %>%
  v_pie(
    aes(x = source, y = generation),
    outerRadius = 0.8,
    innerRadius = 0.5,
    padAngle = 0.6
  ) %>%
  v_specs_tooltip(visible = FALSE) %>%
  v_specs_indicator(
    visible = TRUE,
    trigger = "hover",
    limitRatio = 0.5,
    title = list(
      visible = TRUE,
      autoFit = TRUE,
      fitStrategy = "inscribed",
      style = list(
        fontWeight = "bolder",
        fill = "#888",
        text = JS("datum => datum !== null ? datum.x : ''")
      )
    ),
    content = list(
      list(
        visible = TRUE,
        autoFit = TRUE,
        fitStrategy = "inscribed",
        style = list(
          fontWeight = "bolder",
          fill = "#000",
          text = JS("datum => datum !== null ? Math.round(datum.y) + 'TWh' : ''")
        )
      )
    )
  )
```

v_specs_legend *Set legend options*

Description

Set legend options

Usage

```
v_specs_legend(vc, ..., add = FALSE)
```

Arguments

vc	An htmlwidget created with vchart().
...	Options for the legend, see examples or online documentation.
add	Add the legend to exiting ones or overwrite all previous legends.

Value

A vchart() htmlwidget object.

Examples

```
library(vchartr)
data("mpg", package = "ggplot2")

vchart(table(Class = mpg$class, Year = mpg$year)) %>%
  v_bar(aes(Class, Freq, fill = Year)) %>%
  v_specs_legend(
    title = list(text = "Title", visible = TRUE),
    orient = "right",
    position = "start",
    item = list(focus = TRUE)
  )
```

v_specs_player *Set player options*

Description

Set player options

Usage

```
v_specs_player(vc, ...)
```

Arguments

vc	An htmlwidget created with vchart().
...	Options for the legend, see examples or online documentation.

Value

A vchart() htmlwidget object.

Examples

```
library(vchartr)

world_electricity %>%
  subset(type == "detail") %>%
  vchart() %>%
  v_bar(
    aes(source, generation, player = year)
  )
```

---

v_specs_tooltip *Set tooltip options*

Description

Set tooltip options

Usage

```
v_specs_tooltip(vc, ..., .reset = FALSE)
```

Arguments

vc	An htmlwidget created with vchart().
...	Options for the tooltip, see examples or online documentation.
.reset	Reset previous tooltip configuration before updating.

Value

A vchart() htmlwidget object.

Examples

```
library(vchartr)
data("mpg", package = "ggplot2")

vchart(table(Class = mpg$class, Year = mpg$year)) %>%
  v_bar(aes(Class, Freq, fill = Year)) %>%
  v_specs_tooltip(
    visible = FALSE
  )
```

`v_sunburst` *Create a Sunburst Chart*

Description

Create a Sunburst Chart

Usage

```
v_sunburst(
  vc,
  mapping = NULL,
  data = NULL,
  name = NULL,
  drill = TRUE,
  gap = 5,
  ...,
  serie_id = NULL,
  data_id = NULL
)
```

Arguments

`vc`	A chart initialized with `vchart()`.
`mapping`	Default list of aesthetic mappings to use for chart.
`data`	Default dataset to use for chart. If not already a `data.frame`, it will be coerced to with `as.data.frame`.
`name`	Name for the serie, only used for single serie (no `color`/`fill` aesthetic supplied).
`drill`	Drill-down function switch.
`gap`	Layer gap, supports passing an array to configure layer gaps layer by layer.
`...`	Additional parameters for the serie.
`data_id, serie_id`	ID for the data/serie, can be used to further customize the chart with `v_specs()`.

Value

A vchart() htmlwidget object.

Examples

```
library(vchartr)

# Create a sunburst and auto hide labels
vchart(countries_gdp) %>%
  v_sunburst(
    aes(lvl1 = REGION_UN, lvl2 = SUBREGION, lvl3 = ADMIN, value = GDP_MD),
    gap = 10,
    labelAutoVisible = list(
      enable = TRUE
    ),
    labelLayout = list(
      align = "center",
      rotate = "radial"
    )
  )


# Custom tooltip
vchart(countries_gdp) %>%
  v_sunburst(
    aes(lvl1 = REGION_UN, lvl2 = SUBREGION, lvl3 = ADMIN, value = GDP_MD)
  ) %>%
  v_specs_tooltip(
    mark = list(
      title = list(
        value = JS("val => val?.datum?.map(data => data.name).join(' / ')")
      )
    )
  )


# Custom layout options
vchart(countries_gdp) %>%
  v_sunburst(
    aes(lvl1 = REGION_UN, lvl2 = SUBREGION, lvl3 = ADMIN, value = GDP_MD),
    gap = 0,
    innerRadius = c(0, 0.4, 0.8),
    outerRadius = c(0.3, 0.7, 0.85),
    labelAutoVisible = list(
      enable = TRUE,
      circumference = 1
    ),
    labelLayout = list(
      list(
        align = "center",
        rotate = "tangential",
        offset = 0
```

```
    ),
    NULL,
    list(
      align = "start",
      rotate = "radial",
      offset = 15
    )
  )
) %>%
v_specs(padding = 70)
```

---

v_theme *Theme for Charts*

Description

Theme for Charts

Usage

```
v_theme(
  vc,
  .colorPalette = NULL,
  .backgroundColor = NULL,
  .borderColor = NULL,
  .shadowColor = NULL,
  .hoverBackgroundColor = NULL,
  .sliderRailColor = NULL,
  .sliderHandleColor = NULL,
  .sliderTrackColor = NULL,
  .popupBackgroundColor = NULL,
  .primaryFontColor = NULL,
  .secondaryFontColor = NULL,
  .tertiaryFontColor = NULL,
  .axisLabelFontColor = NULL,
  .disableFontColor = NULL,
  .axisMarkerFontColor = NULL,
  .axisGridColor = NULL,
  .axisDomainColor = NULL,
  .dataZoomHandleStrokeColor = NULL,
  .dataZoomChartColor = NULL,
  .playerControllerColor = NULL,
  .scrollBarSliderColor = NULL,
  .axisMarkerBackgroundColor = NULL,
  .markLabelBackgroundColor = NULL,
  .markLineStrokeColor = NULL,
  .dangerColor = NULL,
  .warningColor = NULL,
```

```
  .successColor = NULL,
  .infoColor = NULL,
  .discreteLegendPagerTextColor = NULL,
  .discreteLegendPagerHandlerColor = NULL,
  .discreteLegendPagerHandlerDisableColor = NULL,
  ...
)
```

Arguments

vc	An htmlwidget created with vchart().
.colorPalette	Vector of colors to use as default.
.backgroundColor	background Color
.borderColor	border Color
.shadowColor	shadow Color
.hoverBackgroundColor	hoverBackground Color
.sliderRailColor	slider Rail Color
.sliderHandleColor	slider Handle Color
.sliderTrackColor	slider Track Color
.popupBackgroundColor	popup Background Color
.primaryFontColor	primary Font Color
.secondaryFontColor	secondary Font Color
.tertiaryFontColor	tertiary Font Color
.axisLabelFontColor	axisLabel Font Color
.disableFontColor	disable Font Color
.axisMarkerFontColor	axis Marker Font Color
.axisGridColor	axis Grid Color
.axisDomainColor	axis Domain Color
.dataZoomHandleStrokeColor	data Zoom Handle Stroke Color
.dataZoomChartColor	data Zoom Chart Color

`.playerControllerColor`
: player Controller Color

`.scrollBarSliderColor`
: scroll Bar Slider Color

`.axisMarkerBackgroundColor`
: axis Marker Background Color

`.markLabelBackgroundColor`
: mark Label Background Color

`.markLineStrokeColor`
: mark Line Stroke Color

`.dangerColor`
: danger Color

`.warningColor`
: warning Color

`.successColor`
: success Color

`.infoColor`
: info Color

`.discreteLegendPagerTextColor`
: discrete Legend Pager Text Color

`.discreteLegendPagerHandlerColor`
: discrete Legend Pager Handler Color

`.discreteLegendPagerHandlerDisableColor`
: discrete Legend Pager Handler Disable Color

`...`
: Other parameters.

Value

A `vchart()` `htmlwidget` object.

Examples

```
library(vchartr)

chart <- subset(
  electricity_mix,
  country %in% c("Germany", "Brazil", "South Korea")
) %>%
  vchart() %>%
  v_bar(aes(country, generation, fill = source))

# Default appearance
chart

# Change background color
chart %>%
  v_theme(.backgroundColor = "#2F2E2F")

# Change default color palette
chart %>%
  v_theme(
    .colorPalette = palette.colors(n = 8, palette = "Okabe-Ito")[-1]
```

```
  )

# Axis grid color
chart %>%
  v_theme(.axisGridColor = "red")
# same as
chart %>%
  v_theme(
    component = list(
      axis = list(
        grid = list(
          style = list(
            # lineWidth = 3, # but more options available
            stroke = "red"
          )
        )
      )
    )
  )
# see https://www.unpkg.com/@visactor/vchart-theme@1.11.6/public/light.json
# for all possibilities
```

---

v_treemap *Create a Treemap Chart*

Description

Create a Treemap Chart

Usage

```
v_treemap(
  vc,
  mapping = NULL,
  data = NULL,
  name = NULL,
  drill = TRUE,
  ...,
  serie_id = NULL,
  data_id = NULL
)
```

Arguments

vc	A chart initialized with vchart().
mapping	Default list of aesthetic mappings to use for chart.
data	Default dataset to use for chart. If not already a data.frame, it will be coerced to with as.data.frame.

name — Name for the serie, only used for single serie (no color/fill aesthetic supplied).

drill — Drill-down function switch.

... — Additional parameters for the serie.

data_id, serie_id — ID for the data/serie, can be used to further customize the chart with v_specs().

Value

A vchart() htmlwidget object.

Examples

```
library(vchartr)

# Basic Treemap Chart
vchart(countries_gdp) %>%
  v_treemap(aes(lvl1 = REGION_UN, lvl2 = ADMIN, value = GDP_MD))

# With labels
vchart(countries_gdp) %>%
  v_treemap(
    aes(lvl1 = REGION_UN, lvl2 = ADMIN, value = GDP_MD),
    label = list(visible = TRUE)
  )

# Show all levels
vchart(countries_gdp) %>%
  v_treemap(
    aes(lvl1 = REGION_UN, lvl2 = ADMIN, value = GDP_MD),
    label = list(visible = TRUE),
    nonLeaf = list(visible = TRUE),
    nonLeafLabel = list(visible = TRUE, position = "top")
  )
```

---

v_venn *Create a Venn Diagram*

Description

Create a Venn Diagram

Usage

```
v_venn(
  vc,
  mapping = NULL,
  data = NULL,
```

```
  name = NULL,
  sets_sep = ",",
  ...,
  serie_id = NULL,
  data_id = NULL
)
```

Arguments

vc	A chart initialized with vchart().
mapping	Default list of aesthetic mappings to use for chart.
data	Default dataset to use for chart. If not already a data.frame, it will be coerced to with as.data.frame.
name	Name for the serie, only used for single serie (no color/fill aesthetic supplied).
sets_sep	Sets separator.
...	Additional parameters for the serie.
data_id, serie_id	ID for the data/serie, can be used to further customize the chart with v_specs().

Value

A vchart() htmlwidget object.

Examples

```
library(vchartr)

# Venn diagram with 2 sets
data.frame(
  sets = c("A", "B", "A,B"),
  value = c(5, 10, 4)
) %>%
  vchart() %>%
  v_venn(aes(sets = sets, value = value))

# with more sets
data.frame(
  sets = c("A", "B", "C", "A,B", "A,C", "B,C", "A,B,C"),
  value = c(8, 10, 12, 4, 4, 4, 2)
) %>%
  vchart() %>%
  v_venn(aes(sets = sets, value = value))


# More complex example
set.seed(20190708)
genes <- paste("gene",1:1000,sep="")
genes <- list(
```

```
  A = sample(genes,300),
  B = sample(genes,525),
  C = sample(genes,440),
  D = sample(genes,350)
)

vchart(stack(genes)) %>%
  v_venn(aes(category = ind, values = values))
```

`v_waterfall` *Create a Waterfall Chart*

Description

Create a Waterfall Chart

Usage

```
v_waterfall(
  vc,
  mapping = NULL,
  data = NULL,
  name = NULL,
  ...,
  serie_id = NULL,
  data_id = NULL
)
```

Arguments

`vc`	A chart initialized with `vchart()`.
`mapping`	Default list of aesthetic mappings to use for chart.
`data`	Default dataset to use for chart. If not already a `data.frame`, it will be coerced to with `as.data.frame`.
`name`	Name for the serie, only used for single serie (no `color`/`fill` aesthetic supplied).
`...`	Additional parameters for the serie.
`data_id, serie_id`	ID for the data/serie, can be used to further customize the chart with `v_specs()`.

Value

A `vchart()` `htmlwidget` object.

Examples

```
library(vchartr)

balance <- data.frame(
  desc = c("Starting Cash",
           "Sales", "Refunds", "Payouts", "Court Losses",
           "Court Wins", "Contracts", "End Cash"),
  amount = c(2000, 3400, -1100, -100, -6600, 3800, 1400, 2800)
)

vchart(balance) %>%
  v_waterfall(aes(x = desc, y = amount))


# With total values and formatting
data.frame(
  x = c("Feb.4", "Feb.11", "Feb.20", "Feb.25", "Mar.4",
        "Mar.11", "Mar.19", "Mar.26", "Apr.1", "Apr.8",
        "Apr.15", "Apr.22", "Apr.29", "May.6", "total"),
  total = c(TRUE, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, TRUE),
  y = c(45L, -5L, 2L, -2L, 2L, 2L, -2L, 1L, 1L, 1L, 2L, 1L, -2L, -1L, NA)
) %>%
  vchart() %>%
  v_waterfall(
    aes(x = x, y = y, total = total),
    stackLabel = list(
      valueType = "absolute",
      formatMethod = JS("text => text + '%'")
    )
  ) %>%
  v_specs_legend(visible = TRUE)
```

---

v_wordcloud *Create a Wordcloud*

Description

Create a Wordcloud

Usage

```
v_wordcloud(
  vc,
  mapping = NULL,
  data = NULL,
  name = NULL,
  ...,
  serie_id = NULL,
  data_id = NULL
)
```

Arguments

vc	A chart initialized with vchart().
mapping	Default list of aesthetic mappings to use for chart.
data	Default dataset to use for chart. If not already a data.frame, it will be coerced to with as.data.frame.
name	Name for the serie, only used for single serie (no color/fill aesthetic supplied).
...	Additional parameters for the serie.
data_id, serie_id	ID for the data/serie, can be used to further customize the chart with v_specs().

Value

A vchart() htmlwidget object.

Examples

```
library(vchartr)

vchart(top_cran_downloads) %>%
  v_wordcloud(aes(word = package, count = count))

vchart(top_cran_downloads) %>%
  v_wordcloud(aes(word = package, count = count, color = package))

vchart(top_cran_downloads) %>%
  v_wordcloud(
    aes(word = package, count = count, color = package),
    wordCloudConfig = list(
      zoomToFit = list(
        enlarge = TRUE,
        fontSizeLimitMax = 30
      )
    )
  )


# Use an image to shape the wordcloud
vchart(top_cran_downloads) %>%
  v_wordcloud(
    aes(word = package, count = count, color = package),
    maskShape = "https://jeroen.github.io/images/Rlogo.png"
  )
```

`world_electricity` *World low carbon & fossil electricity generation 2014 - 2023*

Description

This dataset represents world's electricity generation from low-carbon sources and fossil fuels over the period 2014 - 2023.

Usage

```
world_electricity
```

Format

A data frame with 70 observations and 4 variables:

- `year` : Year
- `source` : Either :
  - `Low carbon` : Electricity generation from low-carbon sources - Low-carbon sources correspond to renewables and nuclear power, that produce significantly less greenhouse-gas emissions than fossil fuels.
  - `Renewables` : Electricity generation from renewables
  - `Nuclear` : Electricity generation from nuclear
  - `Fossil` : Electricity generation from fossil fuels (oil + gas + coal)
  - `Oil` : Electricity generation from fossil fuels
  - `Gas` : Electricity generation from fossil fuels
  - `Coal` : Electricity generation from fossil fuels
- `generation` : Electricity generation in terawatt-hours.
- `type` : Type of source : total or detail.

Source

Our World In Data

Index