Package: dashboardr (via r-universe)

May 26, 2026

Title Build Interactive HTML Dashboards from R

Version 0.6.1

Description Build interactive HTML dashboards from R using a simple, composable grammar. Create visualizations, organize them into pages, and generate complete dashboards with navigation, theming, and interactivity. Powered by Highcharts for charts and Quarto for rendering.

License MIT + file LICENSE

URL https://favstats.github.io/dashboardr/

Encoding UTF-8

SystemRequirements Quarto (>= 1.4) - https://quarto.org/docs/download/

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Imports cli, commonmark, fansi, magrittr, dplyr, forcats, gert, glue, highcharter, htmltools, htmlwidgets, jsonlite, quarto, rlang, tibble, tidyr, tidyselect, usethis, digest

Suggests countrycode, devtools, DT, echarts4r, gapminder, ggalluvial, ggiraph, ggplot2, gssr, gt, haven, httr, knitr, leaflet, plotly, purrr, reactable, rnaturalearth, rmarkdown, rstudioapi, stringr, testthat (>= 3.0.0), withr, yaml

Additional_repositories https://kjhealy.r-universe.dev

VignetteBuilder knitr

Config/testthat/edition 3

Config/pak/sysreqs
cmake git libglpk-dev make libgit2-dev libicu-dev libuv1-dev libxml2-dev libssl-dev libx11-dev

Repository https://favstats.r-universe.dev

Date/Publication 2026-02-22 10:24:37 UTC

RemoteUrl https://github.com/favstats/dashboardr

RemoteRef HEAD

RemoteSha 4dd6e5af72f01a173f497524c9ca9d8db2a4da49

Contents

`+.content_collection` *Combine Content Collections with + Operator*

Description

S3 method for combining content_collection objects using +. Preserves all attributes including lazy loading settings.

Usage

```
## S3 method for class 'content_collection'
e1 + e2
```

Arguments

`e1`	First content_collection
`e2`	Second content_collection

Value

Combined content_collection

`+.viz_collection` *Combine Visualization Collections with + Operator*

Description

S3 method that allows combining two viz_collection objects using the + operator. This is a convenient shorthand for `combine_content`. Preserves all attributes including lazy loading settings.

Usage

```
## S3 method for class 'viz_collection'
e1 + e2
```

Arguments

`e1`	First viz_collection object (left operand).
`e2`	Second viz_collection object (right operand).

Details

The + operator provides an intuitive way to combine visualization collections:

- All visualizations from both collections are merged
- Tabgroup labels are combined (e2 labels take precedence for duplicates)
- Insertion indices are renumbered to maintain proper ordering
- All attributes (lazy loading, etc.) are preserved

Value

A new viz_collection containing all visualizations from both collections, with merged tabgroup labels and renumbered insertion indices.

See Also

combine_content for the underlying function.

Examples

```
## Not run:
# Create two separate visualization collections
viz1 <- create_viz() %>%
  add_viz(type = "histogram", x_var = "age", title = "Age Distribution")

viz2 <- create_viz() %>%
  add_viz(type = "histogram", x_var = "income", title = "Income Distribution")

# Combine using + operator
combined <- viz1 + viz2

# Equivalent to:
combined <- combine_content(viz1, viz2)

## End(Not run)
```

---

add_accordion *Add collapsible accordion/details section*

Description

Add collapsible accordion/details section

Usage

```
add_accordion(
  content,
  title,
  text,
  open = FALSE,
  tabgroup = NULL,
  show_when = NULL
)
```

Arguments

content	A content_collection or viz_collection object
title	Section title
text	Section content
open	Whether section starts open (default: FALSE)
tabgroup	Optional tabgroup for organizing content (character vector for nested tabs)
show_when	One-sided formula controlling conditional display based on input values.

Value

Updated content_collection

---

add_badge *Add a status badge*

Description

Add a status badge

Usage

```
add_badge(content, text, color = "primary", tabgroup = NULL, show_when = NULL)
```

Arguments

content	Content collection object
text	Badge text
color	Badge color (success, warning, danger, info, primary, secondary)
tabgroup	Optional tabgroup for organizing content (character vector for nested tabs)
show_when	One-sided formula controlling conditional display based on input values.

`add_callout` *Add callout box*

Description

Add callout box

Usage

```
add_callout(
  x,
  text,
  type = c("note", "tip", "warning", "caution", "important"),
  title = NULL,
  icon = NULL,
  collapse = FALSE,
  tabgroup = NULL,
  show_when = NULL
)
```

Arguments

`x`	A content_collection, viz_collection, sidebar_container, or page_object
`text`	Callout content
`type`	Callout type (note/tip/warning/caution/important)
`title`	Optional title
`icon`	Optional icon
`collapse`	Whether callout is collapsible
`tabgroup`	Optional tabgroup for organizing content (character vector for nested tabs)
`show_when`	One-sided formula controlling conditional display based on input values.

Value

Updated content_collection

add_callout.page_object *Add a callout to a page*

Description

Add callout boxes directly to a page object.

Usage

```
add_callout.page_object(
  page,
  text,
  type = "note",
  title = NULL,
  tabgroup = NULL,
  show_when = NULL
)
```

Arguments

page	A page_object created by create_page()
text	Callout text content
type	Callout type: "note", "tip", "warning", "important", "caution"
title	Optional callout title
tabgroup	Optional tabgroup
show_when	One-sided formula controlling conditional display based on input values.

Value

The updated page_object

add_card *Add card*

Description

Add card

Usage

```
add_card(
  content,
  text,
  title = NULL,
  footer = NULL,
  tabgroup = NULL,
  show_when = NULL
)
```

Arguments

content	A content_collection or viz_collection object
text	Card content
title	Card title
footer	Optional card footer
tabgroup	Optional tabgroup for organizing content (character vector for nested tabs)
show_when	One-sided formula controlling conditional display based on input values.

Value

Updated content_collection

---

add_code *Add code block*

Description

Add code block

Usage

```
add_code(
  content,
  code,
  language = "r",
  caption = NULL,
  filename = NULL,
  tabgroup = NULL,
  show_when = NULL
)
```

Arguments

content	A content_collection, viz_collection, or page_object
code	Code content
language	Programming language for syntax highlighting
caption	Optional caption
filename	Optional filename to display
tabgroup	Optional tabgroup for organizing content (character vector for nested tabs)
show_when	One-sided formula controlling conditional display based on input values.

Value

Updated object (same type as input)

---

add_content *Add content collection(s) to a page*

Description

Add one or more pre-built content collections (from create_viz/create_content) to a page. Use this when you have complex content built separately.

Usage

```
add_content(page, ...)
```

Arguments

page	A page_object created by create_page()
...	One or more content collections (from create_viz/create_content)

Value

The updated page_object

Examples

```
## Not run:
# Add a single collection
page %>% add_content(my_viz)

# Add multiple collections at once
page %>% add_content(viz1, viz2, viz3)

## End(Not run)
```

add_dashboard_page *Add a page to the dashboard*

Description

Universal function for adding any type of page to the dashboard. Can create landing pages, analysis pages, about pages, or any combination of text and visualizations. All content is markdown-compatible.

Usage

```
add_dashboard_page(
  proj,
  name,
  data = NULL,
  data_path = NULL,
  template = NULL,
  params = list(),
  visualizations = NULL,
  content = NULL,
  text = NULL,
  icon = NULL,
  is_landing_page = FALSE,
  show_in_nav = TRUE,
  tabset_theme = NULL,
  tabset_colors = NULL,
  navbar_align = c("left", "right"),
  overlay = FALSE,
  overlay_theme = c("light", "glass", "dark", "accent"),
  overlay_text = "Loading",
  overlay_duration = 2200,
  lazy_load_charts = NULL,
  lazy_load_margin = NULL,
  lazy_load_tabs = NULL,
  lazy_debug = NULL,
  pagination_separator = NULL,
  time_var = NULL,
  slug = NULL
)
```

Arguments

proj	A dashboard_project object
name	Page display name
data	Optional data frame to save for this page. Can also be a named list of data frames for using multiple datasets: `list(survey = df1, demographics = df2)`

data_path	Path to existing data file (alternative to data parameter). Can also be a named list of file paths for multiple datasets
template	Optional custom template file path
params	Parameters for template substitution
visualizations	Content collection or list of visualization specs
content	Alternative to visualizations - supports content collections
text	Optional markdown text content for the page
icon	Optional iconify icon shortcode (e.g., "ph:users-three")
is_landing_page	Whether this should be the landing page (default: FALSE)
show_in_nav	Whether to show this page in the navbar (default: TRUE). Set to FALSE for pageless dashboards (created with `create_page("")`).
tabset_theme	Optional tabset theme for this page (overrides dashboard-level theme)
tabset_colors	Optional tabset colors for this page (overrides dashboard-level colors)
navbar_align	Position of page in navbar: "left" (default) or "right"
overlay	Whether to show a loading overlay on page load (default: FALSE)
overlay_theme	Theme for loading overlay: "light", "glass", "dark", or "accent" (default: "light")
overlay_text	Text to display in loading overlay (default: "Loading")
overlay_duration	Duration in milliseconds for how long overlay stays visible (default: 2200)
lazy_load_charts	Override dashboard-level lazy loading setting for this page (default: NULL = inherit from dashboard)
lazy_load_margin	Override viewport margin for lazy loading on this page (default: NULL = inherit from dashboard)
lazy_load_tabs	Override tab-aware lazy loading for this page (default: NULL = inherit from dashboard)
lazy_debug	Override debug mode for lazy loading on this page (default: NULL = inherit from dashboard)
pagination_separator	Text to show in pagination navigation (e.g., "of" -> "1 of 3"), default: NULL = inherit from dashboard
time_var	Name of the time/x-axis column in the data (e.g., "year", "decade", "date"). Used by input filters when switching metrics. If NULL (default), the JavaScript will try to auto-detect from common column names (year, decade, time, date).
slug	Optional custom slug for the page filename. If provided, overrides the default name-based slug. Non-alphanumeric characters are replaced with underscores.

Value

The updated dashboard_project object

Examples

```
## Not run:
# Landing page
dashboard <- create_dashboard("test") %>%
  add_page("Welcome", text = "# Welcome\n\nThis is the main page.", is_landing_page = TRUE)

# Analysis page with data and visualizations
dashboard <- dashboard %>%
  add_page("Demographics", data = survey_data, visualizations = demo_viz)

# Text-only about page
dashboard <- dashboard %>%
  add_page("About", text = "# About This Study\n\nThis dashboard shows...")

# Mixed content page
dashboard <- dashboard %>%
  add_page("Results", text = "# Key Findings\n\nHere are the results:",
           visualizations = results_viz, icon = "ph:chart-line")

# Page with explicit time variable for metric switching
dashboard <- dashboard %>%
  add_page("Trends", data = trend_data, visualizations = trend_viz, time_var = "decade")

## End(Not run)
```

---

add_divider *Add horizontal divider*

Description

Add horizontal divider

Usage

```
add_divider(content, style = "default", tabgroup = NULL, show_when = NULL)
```

Arguments

content	A content_collection, viz_collection, or page_object
style	Divider style ("default", "thick", "dashed", "dotted")
tabgroup	Optional tabgroup for organizing content (character vector for nested tabs)
show_when	One-sided formula controlling conditional display based on input values.

Value

Updated object (same type as input)

add_DT *Add DT datatable*

Description

Add DT datatable

Usage

```
add_DT(
  content,
  table_data,
  options = NULL,
  tabgroup = NULL,
  filter_vars = NULL,
  show_when = NULL,
  ...
)
```

Arguments

content	A content_collection object
table_data	A DT datatable object (from DT::datatable()) OR a data frame/matrix (will be auto-converted)
options	List of DT options (only used if passing a data frame)
tabgroup	Optional tabgroup for organizing content (character vector for nested tabs)
filter_vars	Optional character vector of input filter variables to apply to this block.
show_when	One-sided formula controlling conditional display based on input values.
...	Additional arguments passed to DT::datatable() (only used if passing a data frame)

Value

Updated content_collection

Examples

```
## Not run:
# Option 1: Pass a styled DT object
my_dt <- DT::datatable(
  mtcars,
  options = list(pageLength = 10),
  filter = 'top',
  rownames = FALSE
)
```

```
content <- create_content() %>%
  add_DT(my_dt)

# Option 2: Pass a data frame with options
content <- create_content() %>%
  add_DT(mtcars, options = list(pageLength = 5, scrollX = TRUE))

## End(Not run)
```

`add_echarts` *Add an echarts4r chart to the dashboard*

Description

Convenience wrapper around `add_widget` for echarts4r objects.

Usage

```
add_echarts(
  content,
  chart,
  title = NULL,
  height = NULL,
  tabgroup = NULL,
  filter_vars = NULL,
  show_when = NULL
)
```

Arguments

`content`	A content_collection, page_object, or dashboard_project
`chart`	An echarts4r object (created with `echarts4r::e_charts()`)
`title`	Optional title displayed above the chart
`height`	Optional CSS height
`tabgroup`	Optional tabgroup for organizing content
`filter_vars`	Optional character vector of input filter variables to apply to this block.
`show_when`	One-sided formula controlling conditional display based on input values.

Value

Updated content object

`add_filter` *Add a filter control (simplified interface)*

Description

A convenience wrapper around `add_input` for common filtering use cases. Options are automatically derived from the data column specified by `filter_var`. All values are selected by default.

Usage

```
add_filter(
  content,
  filter_var,
  label = NULL,
  type = c("checkbox", "select", "radio"),
  ...
)
```

Arguments

`content`	A content_collection object
`filter_var`	The column name in your data to filter by (quoted or unquoted)
`label`	Optional label for the filter (defaults to the column name)
`type`	Filter type: "checkbox" (default), "select", or "radio"
`...`	Additional arguments passed to `add_input`

Value

Updated content_collection

Examples

```
## Not run:
# Simplest usage - just specify the column!
content <- create_content(data = mydata) %>%
  add_sidebar() %>%
    add_filter(filter_var = "education") %>%
    add_filter(filter_var = "gender") %>%
  end_sidebar() %>%
  add_viz(type = "stackedbar", x_var = "region", stack_var = "outcome")

## End(Not run)
```

add_ggiraph *Add a ggiraph interactive plot to the dashboard*

Description

Convenience wrapper around `add_widget` for ggiraph objects.

Usage

```
add_ggiraph(
  content,
  plot,
  title = NULL,
  height = NULL,
  tabgroup = NULL,
  filter_vars = NULL,
  show_when = NULL
)
```

Arguments

`content`	A content_collection, page_object, or dashboard_project
`plot`	A girafe object (created with `ggiraph::girafe()`)
`title`	Optional title displayed above the plot
`height`	Optional CSS height
`tabgroup`	Optional tabgroup for organizing content
`filter_vars`	Not supported for ggiraph widgets.
`show_when`	One-sided formula controlling conditional display based on input values.

Value

Updated content object

add_ggplot *Add a static ggplot2 plot to the dashboard*

Description

Embed a ggplot2 object directly into a dashboard page. The plot is rendered as a static image via Quarto's built-in knitr graphics device.

Usage

```
add_ggplot(
  content,
  plot,
  title = NULL,
  height = NULL,
  width = NULL,
  tabgroup = NULL,
  show_when = NULL
)
```

Arguments

content	A content_collection, page_object, or dashboard_project
plot	A ggplot2 object (created with ggplot2::ggplot())
title	Optional title displayed above the plot
height	Optional figure height in inches (passed to knitr fig.height)
width	Optional figure width in inches (passed to knitr fig.width)
tabgroup	Optional tabgroup for organizing content
show_when	One-sided formula controlling conditional display based on input values.

Value

Updated content object

---

add_gt *Add gt table*

Description

Add gt table

Usage

```
add_gt(content, gt_object, caption = NULL, tabgroup = NULL, show_when = NULL)
```

Arguments

content	A content_collection object
gt_object	A gt table object (from gt::gt()) OR a data frame (will be auto-converted)
caption	Optional caption
tabgroup	Optional tabgroup for organizing content (character vector for nested tabs)
show_when	One-sided formula controlling conditional display based on input values.

Value

Updated content_collection

Examples

```
## Not run:
# Option 1: Pass a styled gt object
my_table <- gt::gt(mtcars) %>%
  gt::tab_header(title = "Cars") %>%
  gt::fmt_number(columns = everything(), decimals = 1)

content <- create_content() %>%
  add_gt(my_table)

# Option 2: Pass a data frame (auto-converted)
content <- create_content() %>%
  add_gt(mtcars, caption = "Motor Trend Cars")

## End(Not run)
```

---

add_hc *Add a custom highcharter chart*

Description

Add a pre-built highcharter chart to your dashboard. This allows you to create complex, customized highcharter visualizations and include them directly without using dashboardr's viz_* functions.

Usage

```
add_hc(
  content,
  hc_object,
  height = NULL,
  tabgroup = NULL,
  filter_vars = NULL,
  show_when = NULL
)
```

Arguments

content	A content_collection, page_object, or dashboard object
hc_object	A highcharter object created with highcharter::highchart() or hchart()
height	Optional height for the chart (e.g., "400px", "50vh"). If NULL (default), no height is set and highcharter handles its own sizing
tabgroup	Optional tabgroup for organizing content (character vector for nested tabs)
filter_vars	Optional character vector of input filter variables to apply to this block.
show_when	One-sided formula controlling conditional display based on input values.

Value

Updated content object

Examples

```
## Not run:
library(highcharter)

# Create a custom highcharter chart
my_chart <- hchart(mtcars, "scatter", hcaes(x = wt, y = mpg, group = cyl)) %>%
  hc_title(text = "Custom Scatter Plot") %>%
  hc_subtitle(text = "Made with highcharter") %>%
  hc_add_theme(hc_theme_smpl())

# Add it to a page
page <- create_page("Charts") %>%
  add_hc(my_chart) %>%
  add_hc(another_chart, height = "500px", tabgroup = "My Charts")

## End(Not run)
```

---

add_html *Add raw HTML content*

Description

Add raw HTML content

Usage

```
add_html(content, html, tabgroup = NULL, show_when = NULL)
```

Arguments

content	Content collection object
html	Raw HTML string
tabgroup	Optional tabgroup for organizing content (character vector for nested tabs)
show_when	One-sided formula controlling conditional display based on input values.

add_iframe *Add iframe*

Description

Add iframe

Usage

```
add_iframe(
  content,
  src,
  height = "500px",
  width = "100%",
  style = NULL,
  tabgroup = NULL,
  show_when = NULL
)
```

Arguments

content	A content_collection object
src	iframe source URL
height	iframe height (default: "500px")
width	iframe width (default: "100%")
style	Optional inline CSS style string applied to the iframe element (e.g., "border: none; border-radius: 8px;"). Useful for removing borders, adding shadows, or any custom styling.
tabgroup	Optional tabgroup for organizing content (character vector for nested tabs)
show_when	One-sided formula controlling conditional display based on input values.

Value

Updated content_collection

add_image *Add image to content collection (pipeable)*

Description

Adds an image block to a content collection. Can be used standalone or in a pipe. Supports viz_collection as first argument for seamless piping.

Usage

```
add_image(
  content_collection = NULL,
  src,
  alt = NULL,
  caption = NULL,
  width = NULL,
  height = NULL,
  align = c("center", "left", "right"),
  link = NULL,
  class = NULL,
  tabgroup = NULL,
  show_when = NULL
)
```

Arguments

content_collection	A content_collection, viz_collection, or NULL
src	Image source path or URL
alt	Alt text for the image
caption	Optional caption text displayed below the image
width	Optional width (e.g., "300px", "50%", "100%")
height	Optional height (e.g., "200px")
align	Image alignment: "left", "center", "right" (default: "center")
link	Optional URL to link the image to
class	Optional CSS class for custom styling
tabgroup	Optional tabgroup for organizing content (character vector for nested tabs)
show_when	One-sided formula controlling conditional display based on input values.

Value

Updated content_collection object

Examples

```
## Not run:
# Standalone
img <- add_image(src = "logo.png", alt = "Logo")

# Pipe with content
content <- create_content() %>%
  add_text("Welcome!") %>%
  add_image(src = "chart.png", alt = "Chart")

# With tabgroup
```

```
content <- create_content() %>%
  add_image(src = "chart.png", alt = "Chart", tabgroup = "Gallery")

# Pipe directly from viz
content <- create_viz() %>%
  add_viz(type = "bar", x_var = "category") %>%
  add_image(src = "logo.png", alt = "Logo")

## End(Not run)
```

---

add_input *Add an interactive input filter*

Description

Adds an input widget that filters Highcharts visualizations on the page. Supports various input types: dropdowns, checkboxes, radio buttons, switches, sliders, text search, number inputs, and button groups.

Usage

```
add_input(
  content,
  input_id,
  label = NULL,
 type = c("select_multiple", "select_single", "checkbox", "radio", "switch", "slider",
    "text", "number", "button_group", "date", "daterange"),
  filter_var,
  options = NULL,
  options_from = NULL,
  default_selected = NULL,
  placeholder = "Select...",
  width = "300px",
  min = 0,
  max = 100,
  step = 1,
  value = NULL,
  default_value = NULL,
  show_value = TRUE,
  inline = TRUE,
  stacked = FALSE,
  stacked_align = c("center", "left", "right"),
  group_align = c("left", "center", "right"),
  ncol = NULL,
  nrow = NULL,
  columns = NULL,
  toggle_series = NULL,
  override = FALSE,
```

```
  labels = NULL,
  size = c("md", "sm", "lg"),
  help = NULL,
  disabled = FALSE,
  add_all = FALSE,
  add_all_label = "All",
  mt = NULL,
  mr = NULL,
  mb = NULL,
  ml = NULL,
  tabgroup = NULL,
  icons = NULL,
  show_when = NULL,
  .linked_parent_id = NULL,
  .options_by_parent = NULL
)
```

Arguments

content	Content collection object or input_row_container
input_id	Unique ID for this input widget
label	Optional label displayed above the input
type	Input type: "select_multiple" (default), "select_single", "checkbox", "radio", "switch", "slider", "text", "number", or "button_group"
filter_var	The variable name to filter by (matches Highcharts series names). This should match the group_var used in your visualization.
options	Character vector of options to display. If NULL, uses options_from. Required for select, checkbox, radio, and button_group types. Can also be a named list for grouped options in selects (e.g., list("Europe" = c("Germany", "France"), "Asia" = c("China", "Japan"))).
options_from	Column name in page data to auto-populate options from. Only used if options is NULL.
default_selected	Character vector of initially selected values. If NULL, all options are selected by default (for select/checkbox) or first option (for radio/button_group).
placeholder	Placeholder text when nothing is selected (for selects/text)
width	CSS width for the input (default: "300px")
min	Minimum value (for slider/number types)
max	Maximum value (for slider/number types)
step	Step increment (for slider/number types)
value	Initial value (for slider/switch/text/number types)
default_value	Default value for the input (alias for value, used for reset)
show_value	Whether to show current value (for slider, default TRUE)
inline	Whether to display options inline (for checkbox/radio, default TRUE)

stacked	Whether to stack options vertically (for checkbox/radio). Default FALSE.
stacked_align	Alignment when stacked: "center" (default), "left", or "right"
group_align	Alignment for option groups: "left" (default), "center", or "right"
ncol	Number of columns for grid layout of options
nrow	Number of rows for grid layout of options
columns	Column configuration for grid layout
toggle_series	For switch type: name of the series to toggle visibility on/off
override	For switch type: if TRUE, the switch overrides other filters for this series
labels	Custom labels for slider ticks (character vector). The first and last labels are shown at the min/max positions.
size	Size variant: "sm" (small), "md" (medium, default), or "lg" (large)
help	Help text displayed below the input
disabled	Whether the input is disabled (default FALSE)
add_all	Whether to add an "All" option (default FALSE)
add_all_label	Label for the "All" option (default "All")
mt	Margin top (CSS value, e.g., "10px")
mr	Margin right (CSS value)
mb	Margin bottom (CSS value)
ml	Margin left (CSS value)
tabgroup	Optional tabgroup for organizing content
icons	Optional character vector of Iconify icon names (e.g., "ph:calendar", "ph:users-three"). Must be same length as options. When provided, icons are rendered before option text for radio and button_group types.
show_when	Optional one-sided formula for conditional visibility of this input (e.g., ~ demo == "By Age"). Wraps the input in a show-when container so it only appears when the condition is met.
.linked_parent_id	Internal. ID of linked parent input for cascading inputs
.options_by_parent	Internal. Named list mapping parent values to child options

Value

Updated content_collection or input_row_container

Examples

```
## Not run:
# Dropdown (multi-select)
content <- create_content() %>%
  add_input(
    input_id = "country_filter",
    label = "Select Countries:",
```

```
    type = "select_multiple",
    filter_var = "country",
    options_from = "country",
    help = "Select one or more countries to compare"
  )

# Grouped select options
content <- create_content() %>%
  add_input(
    input_id = "country_filter",
    label = "Select Countries:",
    type = "select_multiple",
    filter_var = "country",
    options = list(
      "Europe" = c("Germany", "France", "UK"),
      "Asia" = c("China", "Japan", "India")
    )
  )

# Checkbox group
content <- create_content() %>%
  add_input(
    input_id = "metrics",
    label = "Metrics:",
    type = "checkbox",
    filter_var = "metric",
    options = c("Revenue", "Users", "Growth"),
    inline = TRUE
  )

# Radio buttons
content <- create_content() %>%
  add_input(
    input_id = "chart_type",
    label = "Chart Type:",
    type = "radio",
    filter_var = "chart_type",
    options = c("Line", "Bar", "Area")
  )

# Switch/toggle to show/hide a specific series
content <- create_content() %>%
  add_input(
    input_id = "show_average",
    label = "Show Global Average",
    type = "switch",
    filter_var = "country",
    toggle_series = "Global Average",  # Name of the series to toggle
    override = TRUE,                   # Don't let other filters hide this series
    value = TRUE                       # Start with switch ON
  )

# Slider with custom labels
```

```
content <- create_content() %>%
  add_input(
    input_id = "decade_filter",
    label = "Decade:",
    type = "slider",
    filter_var = "decade",
    min = 1,
    max = 6,
    step = 1,
    value = 1,
    labels = c("1970s", "1980s", "1990s", "2000s", "2010s", "2020s")
  )

# Text search input
content <- create_content() %>%
  add_input(
    input_id = "search",
    label = "Search:",
    type = "text",
    filter_var = "name",
    placeholder = "Type to search...",
    size = "lg"
  )

# Button group (segmented control)
content <- create_content() %>%
  add_input(
    input_id = "period",
    label = "Time Period:",
    type = "button_group",
    filter_var = "period",
    options = c("Day", "Week", "Month", "Year")
  )

## End(Not run)
```

---

add_input_row *Start an input row*

Description

Creates a container for input widgets that will be displayed in a horizontal row. The inputs will wrap responsively on smaller screens. Use with end_input_row().

Usage

```
add_input_row(
  content,
  tabgroup = NULL,
  style = c("boxed", "inline"),
```

```
  align = c("center", "left", "right")
)
```

Arguments

content	Content collection object
tabgroup	Optional tabgroup for organizing content. Use this to place the input row inside a specific tab (e.g., "trends" or "trends/Female Authorship")
style	Visual style: "boxed" (default, with background and border) or "inline" (compact, transparent background)
align	Alignment: "center" (default), "left", or "right"

Value

An input_row_container for piping

Examples

```
## Not run:
content <- create_content() %>%
  add_input_row() %>%
    add_input(input_id = "country", filter_var = "country", options_from = "country") %>%
    add_input(input_id = "metric", filter_var = "metric", options_from = "metric") %>%
  end_input_row()

# Place inputs inside a tabgroup
content <- create_content() %>%
  add_input_row(tabgroup = "trends", style = "inline") %>%
    add_input(input_id = "country", filter_var = "country", options = c("A", "B")) %>%
  end_input_row()

## End(Not run)
```

---

add_layout_column *Start a manual layout column*

Description

Creates a column container for explicit Quarto dashboard layout control. Use with `add_layout_row()` and `end_layout_column()`.

Usage

```
add_layout_column(
  content,
  width = NULL,
  class = NULL,
  tabgroup = NULL,
  show_when = NULL
)
```

Arguments

content	A content_collection or page_object.
width	Optional Quarto column width value.
class	Optional CSS class for the column.
tabgroup	Optional tabgroup metadata (reserved for future use).
show_when	Optional one-sided formula controlling visibility.

Value

A layout_column_container for piping.

Examples

```
## Not run:
content <- create_content() %>%
  add_layout_column(width = 60) %>%
  add_layout_row() %>%
    add_text("### Row content") %>%
  end_layout_row() %>%
end_layout_column()

## End(Not run)
```

---

add_layout_row *Start a manual layout row inside a layout column*

Description

Start a manual layout row inside a layout column

Usage

```
add_layout_row(
  column_container,
  class = NULL,
  style = NULL,
  tabgroup = NULL,
  show_when = NULL
)
```

Arguments

column_container	A layout_column_container created by add_layout_column().
class	Optional CSS class for the row.

style	Optional inline CSS style string applied to the row wrapper. In non-dashboard mode this is added to the layout-ncol div; in dashboard mode it is added to the ### Row attributes.
tabgroup	Optional tabgroup metadata (reserved for future use).
show_when	Optional one-sided formula controlling visibility.

Value

A layout_row_container for piping.

---

add_leaflet *Add a leaflet map to the dashboard*

Description

Convenience wrapper around add_widget for leaflet objects.

Usage

```
add_leaflet(
  content,
  map,
  title = NULL,
  height = NULL,
  tabgroup = NULL,
  filter_vars = NULL,
  show_when = NULL
)
```

Arguments

content	A content_collection, page_object, or dashboard_project
map	A leaflet object (created with leaflet::leaflet())
title	Optional title displayed above the map
height	Optional CSS height
tabgroup	Optional tabgroup for organizing content
filter_vars	Optional character vector of input filter variables to apply to this block.
show_when	One-sided formula controlling conditional display based on input values.

Value

Updated content object

`add_linked_inputs` *Add linked parent-child inputs (cascading dropdowns)*

Description

Creates two linked select inputs where the child's available options depend on the parent's current selection. Can be used inside a sidebar (after `add_sidebar()`), inside an input row (after `add_input_row()`), or directly in a content collection.

Usage

```
add_linked_inputs(x, parent, child, type = "select")
```

Arguments

`x`	A sidebar_container, input_row_container, or content_collection.
`parent`	List with: `id`, `label`, `options`; optionally `default_selected`, `filter_var`.
`child`	List with: `id`, `label`, `options_by_parent` (named list mapping each parent value to a character vector of child options); optionally `filter_var`.
`type`	Input type for parent: `"select"` (default) or `"radio"`.

Value

The modified container for piping.

Examples

```
## Not run:
# Inside a sidebar
add_sidebar() %>%
  add_linked_inputs(
    parent = list(id = "dimension", label = "Dimension",
                  options = c("AI", "Safety", "Digital Health")),
    child = list(id = "question", label = "Question",
                 options_by_parent = list(
                   "AI" = c("Overall", "Using AI Tools"),
                   "Safety" = c("Overall", "Passwords", "Phishing"),
                   "Digital Health" = c("Overall", "Screen Time")
                 ))
  ) %>%
  end_sidebar()

# Inside main content (no sidebar needed)
create_content(data = my_data) %>%
  add_linked_inputs(
    parent = list(id = "category", label = "Category",
                  options = c("Overall", "By Age", "By Gender")),
    child = list(id = "value", label = "Value",
```

```
                  options_by_parent = list(
                    "Overall" = c("All"),
                    "By Age" = c("18-34", "35-54", "55+"),
                    "By Gender" = c("Male", "Female")
                  ))
  )

## End(Not run)
```

`add_metric` *Add a metric/value box*

Description

Add a metric/value box

Usage

```
add_metric(
  content,
  value,
  title,
  icon = NULL,
  color = NULL,
  bg_color = NULL,
  text_color = NULL,
  gradient = TRUE,
  gradient_intensity = 0.45,
  value_prefix = NULL,
  value_suffix = NULL,
  border_radius = NULL,
  subtitle = NULL,
  tabgroup = NULL,
  show_when = NULL,
  aria_label = NULL
)
```

Arguments

`content`	Content collection object
`value`	The metric value
`title`	Metric title
`icon`	Optional icon
`color`	Optional color theme
`bg_color`	Optional background color (e.g. "#3498db").
`text_color`	Optional text color (e.g. "#ffffff").

value_prefix	Optional string prepended to the displayed value.
value_suffix	Optional string appended to the displayed value.
border_radius	Optional CSS border-radius (e.g. "12px", "0").
subtitle	Optional subtitle text
tabgroup	Optional tabgroup for organizing content (character vector for nested tabs)
show_when	One-sided formula controlling conditional display based on input values.
aria_label	Optional ARIA label for accessibility.

---

add_modal *Add Modal to Content Collection (Pipeable)*

Description

Adds a modal definition to your content collection. Use markdown links with {.modal-link} class to trigger the modal.

Usage

```
add_modal(
  x,
  modal_id,
  title = NULL,
  modal_content = NULL,
  image = NULL,
  image_width = "100%",
  ...
)
```

Arguments

x	A content_collection, viz_collection, or page_object to add modal to
modal_id	Unique ID for this modal (used in markdown link)
title	Modal title (optional)
modal_content	Text content - can be plain text, HTML, or data.frame
image	Optional image URL or path
image_width	Width of the image (default "100%"). Can be percentage ("70%") or pixels ("500px")
...	Additional content (data.frames will be converted to tables)

Value

Updated content_collection with modal added

Examples

```
## Not run:
# Pipeable syntax (RECOMMENDED)
content <- create_content() %>%
  add_text("## Results") %>%
  add_text("[View details](#details){.modal-link}") %>%
  add_modal(
    modal_id = "details",
    title = "Full Results",
    modal_content = "Detailed analysis here..."
  )

# With image (custom width)
content <- create_viz() %>%
  add_viz(type = "column", x_var = "x", y_var = "y") %>%
  add_modal(
    modal_id = "chart-details",
    title = "Chart Details",
    image = "chart.png",
    image_width = "70%",  # Control image width
    modal_content = "This chart shows..."
  )

# With data.frame (auto-converts to table)
content <- create_content() %>%
  add_text("[View data](#data){.modal-link}") %>%
  add_modal(
    modal_id = "data",
    title = "Raw Data",
    modal_content = head(mtcars, 10)
  )

# Works with page objects too
page <- create_page("Results", data = my_data, type = "bar") %>%
  add_text("[View details](#info){.modal-link}") %>%
  add_modal(
    modal_id = "info",
    title = "More Info",
    modal_content = "Additional details..."
  )

## End(Not run)
```

---

add_navbar_element *Add a custom navbar element to dashboard*

Description

Adds a custom link or element to the navbar. Can include text, icons, and external links. Elements are added to the right side of the navbar by default but can be positioned left.

Usage

```
add_navbar_element(
  proj,
  text = NULL,
  icon = NULL,
  href,
  align = c("right", "left")
)
```

Arguments

proj	Dashboard project object from create_dashboard()
text	Display text for the element (optional if icon provided)
icon	Iconify icon (e.g., "ph:lightning-fill") (optional)
href	Hyperlink URL (required)
align	Position in navbar: "left" or "right" (default: "right")

Value

Modified dashboard project object

Examples

```
## Not run:
# Add a "Powered by X" link with icon
dashboard <- create_dashboard("my_dashboard", "My Dashboard") %>%
  add_page("Home", text = "# Welcome", is_landing_page = TRUE) %>%
  add_navbar_element(
    text = "Powered by X",
    icon = "ph:lightning-fill",
    href = "https://example.com",
    align = "right"
  )

# Add multiple elements
dashboard <- create_dashboard("my_dashboard", "Dashboard") %>%
  add_page("Home", ...) %>%
  add_navbar_element(
    text = "Documentation",
    icon = "ph:book-open",
    href = "https://docs.example.com"
  ) %>%
  add_navbar_element(
    text = "Sponsor",
    icon = "ph:star-fill",
    href = "https://sponsor.com"
  )

# Icon only (no text)
dashboard %>%
```

```
  add_navbar_element(
    icon = "ph:github-logo",
    href = "https://github.com/user/repo"
  )

## End(Not run)
```

---

add_page *Add Page to Dashboard (Alias)*

Description

Convenient alias for add_dashboard_page. Adds a new page to a dashboard project.

Usage

```
add_page(
  proj,
  name,
  data = NULL,
  data_path = NULL,
  template = NULL,
  params = list(),
  visualizations = NULL,
  content = NULL,
  text = NULL,
  icon = NULL,
  is_landing_page = FALSE,
  show_in_nav = TRUE,
  tabset_theme = NULL,
  tabset_colors = NULL,
  navbar_align = c("left", "right"),
  overlay = FALSE,
  overlay_theme = c("light", "glass", "dark", "accent"),
  overlay_text = "Loading",
  overlay_duration = 2200,
  lazy_load_charts = NULL,
  lazy_load_margin = NULL,
  lazy_load_tabs = NULL,
  lazy_debug = NULL,
  pagination_separator = NULL,
  time_var = NULL,
  slug = NULL
)
```

Arguments

proj	A dashboard_project object
name	Page display name
data	Optional data frame to save for this page. Can also be a named list of data frames for using multiple datasets: `list(survey = df1, demographics = df2)`
data_path	Path to existing data file (alternative to data parameter). Can also be a named list of file paths for multiple datasets
template	Optional custom template file path
params	Parameters for template substitution
visualizations	Content collection or list of visualization specs
content	Alternative to visualizations - supports content collections
text	Optional markdown text content for the page
icon	Optional iconify icon shortcode (e.g., "ph:users-three")
is_landing_page	Whether this should be the landing page (default: FALSE)
show_in_nav	Whether to show this page in the navbar (default: TRUE). Set to FALSE for pageless dashboards (created with `create_page("")`).
tabset_theme	Optional tabset theme for this page (overrides dashboard-level theme)
tabset_colors	Optional tabset colors for this page (overrides dashboard-level colors)
navbar_align	Position of page in navbar: "left" (default) or "right"
overlay	Whether to show a loading overlay on page load (default: FALSE)
overlay_theme	Theme for loading overlay: "light", "glass", "dark", or "accent" (default: "light")
overlay_text	Text to display in loading overlay (default: "Loading")
overlay_duration	Duration in milliseconds for how long overlay stays visible (default: 2200)
lazy_load_charts	Override dashboard-level lazy loading setting for this page (default: NULL = inherit from dashboard)
lazy_load_margin	Override viewport margin for lazy loading on this page (default: NULL = inherit from dashboard)
lazy_load_tabs	Override tab-aware lazy loading for this page (default: NULL = inherit from dashboard)
lazy_debug	Override debug mode for lazy loading on this page (default: NULL = inherit from dashboard)
pagination_separator	Text to show in pagination navigation (e.g., "of" -> "1 of 3"), default: NULL = inherit from dashboard
time_var	Name of the time/x-axis column in the data (e.g., "year", "decade", "date"). Used by input filters when switching metrics. If NULL (default), the JavaScript will try to auto-detect from common column names (year, decade, time, date).
slug	Optional custom slug for the page filename. If provided, overrides the default name-based slug. Non-alphanumeric characters are replaced with underscores.

Value

Modified dashboard project with the new page added.

See Also

add_dashboard_page for full parameter documentation.

---

add_pages *Add multiple pages to a dashboard*

Description

Adds one or more page objects to a dashboard.

Usage

```
add_pages(proj, ...)
```

Arguments

proj	A dashboard_project object
...	One or more page_objects to add

Value

The updated dashboard_project object

Examples

```
## Not run:
home <- create_page("Home", is_landing_page = TRUE) %>%
  add_text("# Welcome!")

analysis <- create_page("Analysis", data = gss, type = "bar") %>%
  add_viz(x_var = "degree", title = "Education") %>%
  add_viz(x_var = "race", title = "Race")

create_dashboard(title = "My Dashboard") %>%
  add_pages(home, analysis) %>%
  generate_dashboard()

## End(Not run)
```

add_pagination *Add pagination break to visualization collection*

Description

Insert a pagination marker that splits the visualization collection into separate HTML pages. Each section will be rendered as its own page file (e.g., analysis.html, analysis_p2.html, analysis_p3.html) with automatic Previous/Next navigation between them.

Usage

```
add_pagination(viz_collection, position = NULL)

add_pagination.page_object(viz_collection, position = NULL)
```

Arguments

viz_collection	A viz_collection object
position	Position for pagination controls: "bottom" (sticky at bottom), "top" (inline with page title), "both" (top and bottom), or NULL (default - uses dashboard-level setting from create_dashboard). Per-page override of the dashboard default.

Details

This provides TRUE performance benefits - each page loads independently, dramatically reducing initial render time and file size for large dashboards.

Value

Updated viz_collection object

Examples

```
## Not run:
# Split 150 charts into 3 pages of 50 each
vizzes <- create_viz()

# Page 1: Charts 1-50
for (i in 1:50) vizzes <- vizzes %>% add_viz(type = "bar", x_var = "cyl")

vizzes <- vizzes %>% add_pagination()  # Split here

# Page 2: Charts 51-100
for (i in 51:100) vizzes <- vizzes %>% add_viz(type = "bar", x_var = "gear")

vizzes <- vizzes %>% add_pagination()  # Split here

# Page 3: Charts 101-150
for (i in 101:150) vizzes <- vizzes %>% add_viz(type = "bar", x_var = "hp")
```

```
# Use in dashboard
dashboard %>%
  add_page("Analysis", visualizations = vizzes)

## End(Not run)
```

`add_plotly` *Add a plotly chart to the dashboard*

Description

Convenience wrapper around `add_widget` for plotly objects.

Usage

```
add_plotly(
  content,
  plot,
  title = NULL,
  height = NULL,
  tabgroup = NULL,
  filter_vars = NULL,
  show_when = NULL
)
```

Arguments

`content`	A content_collection, page_object, or dashboard_project
`plot`	A plotly object (created with `plotly::plot_ly()` or `plotly::ggplotly()`)
`title`	Optional title displayed above the chart
`height`	Optional CSS height
`tabgroup`	Optional tabgroup for organizing content
`filter_vars`	Optional character vector of input filter variables to apply to this block.
`show_when`	One-sided formula controlling conditional display based on input values.

Value

Updated content object

add_powered_by_dashboardr *Add "Powered by dashboardr" branding to footer*

Description

Adds a subtle, sleek "Powered by dashboardr" badge with logo to the bottom-right of the page footer. Integrates seamlessly with existing footer content.

Usage

```
add_powered_by_dashboardr(dashboard, size = "small", style = "default")
```

Arguments

dashboard	A dashboard project created with create_dashboard
size	Size of the branding: "small" (default), "medium", or "large"
style	Style variant: "default", "minimal", or "badge"

Value

Updated dashboard project with dashboardr branding in footer

Examples

```
## Not run:
dashboard <- create_dashboard("my_dash", "My Dashboard") %>%
  add_page(name = "Home", text = "Welcome!") %>%
  add_powered_by_dashboardr()

# With custom size
dashboard <- create_dashboard("my_dash") %>%
  add_powered_by_dashboardr(size = "medium", style = "badge")

## End(Not run)
```

add_quote *Add a blockquote*

Description

Add a blockquote

Usage

```
add_quote(
  content,
  quote,
  attribution = NULL,
  cite = NULL,
  tabgroup = NULL,
  show_when = NULL
)
```

Arguments

content	Content collection object
quote	Quote text
attribution	Optional attribution/source
cite	Optional citation URL
tabgroup	Optional tabgroup for organizing content (character vector for nested tabs)
show_when	One-sided formula controlling conditional display based on input values.

---

add_reactable *Add reactable table*

Description

Add reactable table

Usage

```
add_reactable(
  content,
  reactable_object,
  tabgroup = NULL,
  filter_vars = NULL,
  show_when = NULL
)
```

Arguments

content	A content_collection object
reactable_object	A reactable object (from reactable::reactable()) OR a data frame (will be auto-converted)
tabgroup	Optional tabgroup for organizing content (character vector for nested tabs)
filter_vars	Optional character vector of input filter variables to apply to this block.
show_when	One-sided formula controlling conditional display based on input values.

Value

Updated content_collection

Examples

```
## Not run:
# Option 1: Pass a styled reactable object
my_table <- reactable::reactable(
  mtcars,
  columns = list(mpg = reactable::colDef(name = "MPG")),
  searchable = TRUE,
  striped = TRUE
)

content <- create_content() %>%
  add_reactable(my_table)

# Option 2: Pass a data frame (auto-converted with defaults)
content <- create_content() %>%
  add_reactable(mtcars)

## End(Not run)
```

---

add_reset_button *Add a reset button to reset filters*

Description

Creates a button that resets specified inputs to their default values. Can be used inside a sidebar pipeline (pass a sidebar_container as the first argument) or standalone to generate the raw HTML.

Usage

```
add_reset_button(
  sidebar_container = NULL,
  targets = NULL,
  label = "Reset Filters",
  size = "md"
)
```

Arguments

sidebar_container	A sidebar_container (created by add_sidebar()), or NULL for standalone HTML output.
targets	Character vector of input IDs to reset, or NULL for all
label	Button label
size	Size variant: "sm", "md", or "lg"

Value

Modified sidebar_container when piped, or HTML string when standalone.

`add_sidebar` *Add a sidebar to a page*

Description

Creates a sidebar container that can hold inputs, text, images, and other content. Use with end_sidebar() to close the sidebar and return to main content.

Usage

```
add_sidebar(
  content,
  width = "250px",
  position = c("left", "right"),
  title = NULL,
  background = NULL,
  padding = NULL,
  border = TRUE,
  open = TRUE,
  class = NULL
)
```

Arguments

`content`	Content collection or page_object
`width`	CSS width for sidebar (default "250px")
`position`	Sidebar position: "left" (default) or "right"
`title`	Optional title displayed at top of sidebar
`background`	Background color (CSS color value, e.g., "#f8f9fa", "white", "transparent")
`padding`	Padding inside the sidebar (CSS value, e.g., "1rem", "20px")
`border`	Show border on sidebar edge. TRUE (default), FALSE, or CSS border value
`open`	Whether sidebar starts open (default TRUE). Set FALSE to start collapsed.
`class`	Additional CSS class(es) to add to the sidebar

Details

Sidebars are collapsible vertical panels that appear alongside the main content. They're ideal for placing filter controls, navigation, or supplementary information.

Value

A sidebar_container for piping

Important - Heading Levels

When using a sidebar, the page is rendered in Quarto dashboard format where heading levels have special meaning:

- ## creates new rows/columns (avoid in main content)
- ### creates cards/sections (safe to use)

To avoid layout issues, use ### headings or plain text in the main content area after the sidebar. For advanced layouts, prefer explicit `add_layout_column()` / `add_layout_row()` APIs instead of heading-based layout shaping.

Examples

```
## Not run:
# Basic sidebar with filters
content <- create_content() %>%
  add_sidebar(width = "300px") %>%
    add_text("### Filters") %>%
    add_input(input_id = "country", filter_var = "country", options = countries) %>%
    add_divider() %>%
    add_image(src = "logo.png") %>%
  end_sidebar() %>%
  add_viz(viz_bar(...))

# Right-positioned sidebar
content <- create_content() %>%
  add_sidebar(position = "right", title = "Options") %>%
    add_input(input_id = "metric", filter_var = "metric", type = "radio",
              options = c("Revenue", "Users", "Growth")) %>%
  end_sidebar() %>%
  add_viz(viz_timeline(...))

# Styled sidebar with custom background and no border
content <- create_content() %>%
  add_sidebar(width = "300px", background = "#f8f9fa",
              padding = "1.5rem", border = FALSE) %>%
    add_text("### Settings") %>%
  end_sidebar()

## End(Not run)
```

---

add_spacer *Add vertical spacer*

Description

Add vertical spacer

Usage

```
add_spacer(content, height = "2rem", tabgroup = NULL, show_when = NULL)
```

Arguments

content	A content_collection, viz_collection, or page_object
height	Height (CSS unit, e.g. "2rem", "50px")
tabgroup	Optional tabgroup for organizing content (character vector for nested tabs)
show_when	One-sided formula controlling conditional display based on input values.

Value

Updated object (same type as input)

---

add_table *Add generic table (data frame)*

Description

Add generic table (data frame)

Usage

```
add_table(
  content,
  table_object,
  caption = NULL,
  tabgroup = NULL,
  filter_vars = NULL,
  show_when = NULL
)
```

Arguments

content	A content_collection object
table_object	A data frame or tibble
caption	Optional caption
tabgroup	Optional tabgroup for organizing content (character vector for nested tabs)
filter_vars	Optional character vector of input filter variables to apply to this block.
show_when	One-sided formula controlling conditional display based on input values.

Value

Updated content_collection

add_text *Add text to content collection (pipeable)*

Description

Adds a text block to a content collection. Can be used standalone or in a pipe. Supports viz_collection as first argument for seamless piping.

Usage

```
add_text(x = NULL, text, ..., tabgroup = NULL, show_when = NULL)
```

Arguments

x	A content_collection, viz_collection, sidebar_container, page_object, or NULL
text	Markdown text content (can be multi-line)
...	Additional text lines (will be combined with newlines)
tabgroup	Optional tabgroup for organizing content (character vector for nested tabs)
show_when	One-sided formula controlling conditional display based on input values.

Value

Updated content_collection object

Examples

```
## Not run:
# Standalone
text_block <- add_text("# Welcome")

# Pipe with content
content <- create_content() %>%
  add_text("## Introduction")

# With tabgroup
content <- create_content() %>%
  add_text("## Section 1", tabgroup = "Overview")

# Pipe directly from viz
content <- create_viz() %>%
  add_viz(type = "histogram", x_var = "age") %>%
  add_text("Analysis complete")

## End(Not run)
```

add_text.page_object *Add text to a page*

Description

Add markdown text content directly to a page object.

Usage

```
add_text.page_object(page, text, ..., tabgroup = NULL, show_when = NULL)
```

Arguments

page	A page_object created by create_page()
text	First line of text
...	Additional text lines
tabgroup	Optional tabgroup for the text
show_when	One-sided formula controlling conditional display based on input values.

Value

The updated page_object

add_value_box *Add a custom styled value box*

Description

Creates a modern value box with optional logo, custom background color, and optional collapsible description. Perfect for displaying KPIs and metrics with additional context.

Usage

```
add_value_box(
  content,
  title,
  value,
  logo_url = NULL,
  logo_text = NULL,
  bg_color = "#2c3e50",
  description = NULL,
  description_title = "About this source",
  tabgroup = NULL,
  show_when = NULL,
  aria_label = NULL
)
```

Arguments

content	Content collection object or value_box_row_container
title	Box title (small text above value)
value	Main value to display (large text)
logo_url	Optional URL or path to logo image
logo_text	Optional text to display as logo (if no logo_url)
bg_color	Background color (hex code), default "#2c3e50"
description	Optional collapsible description text (markdown supported)
description_title	Title for collapsible section, default "About this source"
tabgroup	Optional tabgroup for organizing content (character vector for nested tabs)
show_when	One-sided formula controlling conditional display based on input values.
aria_label	Optional ARIA label for accessibility.

Details

Can be used standalone or within a value box row:

- Standalone: create_content() %>% add_value_box(...)
- In row: create_content() %>% add_value_box_row() %>% add_value_box(...) %>% add_value_box(...)

Examples

```
## Not run:
# Standalone value box
content <- create_content() %>%
  add_value_box(
    title = "Total Revenue",
    value = "EUR 1,234,567",
    logo_text = "$",
    bg_color = "#2E86AB"
  )

# Row of value boxes (pipeable!)
content <- create_content() %>%
  add_value_box_row() %>%
    add_value_box(title = "Users", value = "1,234") %>%
    add_value_box(title = "Revenue", value = "EUR 56K")

## End(Not run)
```

add_value_box_row *Start a value box row*

Description

Creates a container for value boxes that will be displayed in a horizontal row. The boxes will wrap responsively on smaller screens. Use pipeable syntax with end_value_box_row():

Usage

```
add_value_box_row(content, tabgroup = NULL, show_when = NULL)
```

Arguments

content	Content collection object
tabgroup	Optional tabgroup for organizing content (character vector for nested tabs)
show_when	One-sided formula controlling conditional display based on input values.

Examples

```
## Not run:
content <- create_content() %>%
  add_value_box_row() %>%
    add_value_box(title = "Users", value = "1,234", bg_color = "#2E86AB") %>%
    add_value_box(title = "Revenue", value = "EUR 56K", bg_color = "#F18F01") %>%
    add_value_box(title = "Growth", value = "+23%", bg_color = "#A23B72") %>%
  end_value_box_row()

## End(Not run)
```

add_video *Add video*

Description

Add video

Usage

```
add_video(
  content,
  src,
  caption = NULL,
  width = NULL,
  height = NULL,
  tabgroup = NULL,
  show_when = NULL
)
```

Arguments

content	A content_collection object
src	Video source URL or path
caption	Optional caption
width	Optional width
height	Optional height
tabgroup	Optional tabgroup for organizing content (character vector for nested tabs)
show_when	One-sided formula controlling conditional display based on input values.

Value

Updated content_collection

---

add_viz *Add a visualization to the collection*

Description

Adds a single visualization specification to an existing collection. Visualizations with the same tabgroup value will be organized into tabs on the generated page. Supports nested tabsets through hierarchy notation.

Usage

```
add_viz(
  x,
  type = NULL,
  ...,
  tabgroup = NULL,
  title = NULL,
  title_tabset = NULL,
  text = NULL,
  icon = NULL,
  text_position = NULL,
  text_before_tabset = NULL,
  text_after_tabset = NULL,
  text_before_viz = NULL,
  text_after_viz = NULL,
  height = NULL,
  filter = NULL,
  data = NULL,
  drop_na_vars = FALSE,
  show_when = NULL
)
```

Arguments

x	A viz_collection or page_object to add visualization to
type	Visualization type: "stackedbar", "heatmap", "histogram", "timeline", "scatter", "bar"
...	Additional parameters passed to the visualization function
tabgroup	Optional group ID for organizing related visualizations. Supports: • Simple string: `"demographics"` for a single tab group • Slash notation: `"demographics/details"` or `"demographics/details/regional"` for nested tabs • Named numeric vector: `c("1" = "demographics", "2" = "details", "3" = "regional")` for explicit hierarchy
title	Display title for the visualization (shown above the chart)
title_tabset	Optional tab label. If NULL, uses `title` for the tab label. Use this when you want a short tab name but a longer, descriptive visualization title.
text	Optional markdown text to display above the visualization
icon	Optional iconify icon shortcode for the visualization
text_position	Position of text relative to visualization ("above" or "below")
text_before_tabset	Optional markdown text to display before the tabset
text_after_tabset	Optional markdown text to display after the tabset
text_before_viz	Optional markdown text to display before the visualization
text_after_viz	Optional markdown text to display after the visualization
height	Optional height in pixels for highcharter visualizations (numeric value)
filter	Optional filter expression to subset data for this visualization. Use formula syntax: `~ condition`. Examples: `~ wave == 1`, `~ age > 18`, `~ wave %in% c(1, 2, 3)`
data	Optional dataset name when using multiple datasets. Can be: • NULL: Uses default dataset (or only dataset if single) • String: Name of dataset from named list (e.g., "survey", "demographics")
drop_na_vars	Whether to drop NA values from variables (default FALSE)
show_when	Conditional display expression for sidebar-driven visibility

Value

The updated viz_collection object

Examples

```
## Not run:
# Simple tabgroup
page1_viz <- create_viz() %>%
  add_viz(type = "stackedbar", x_var = "education", stack_var = "gender",
          title = "Education by Gender", tabgroup = "demographics")

# Nested tabgroups using slash notation
page2_viz <- create_viz() %>%
  add_viz(type = "stackedbar", title = "Overview",
          tabgroup = "demographics") %>%
  add_viz(type = "stackedbar", title = "Details",
          tabgroup = "demographics/details")

# Nested tabgroups using named numeric vector
page3_viz <- create_viz() %>%
  add_viz(type = "stackedbar", title = "Regional Details",
          tabgroup = c("1" = "demographics", "2" = "details", "3" = "regional"))

# Filter data per visualization
page4_viz <- create_viz() %>%
  add_viz(type = "histogram", x_var = "response",
          title = "Wave 1", filter = ~ wave == 1) %>%
  add_viz(type = "histogram", x_var = "response",
          title = "Wave 2", filter = ~ wave == 2) %>%
  add_viz(type = "histogram", x_var = "response",
          title = "All Waves", filter = ~ wave %in% c(1, 2, 3))

# Multiple datasets
page5_viz <- create_viz() %>%
  add_viz(type = "histogram", x_var = "age", data = "demographics") %>%
  add_viz(type = "histogram", x_var = "response", data = "survey") %>%
  add_viz(type = "histogram", x_var = "outcome", data = "outcomes")

# Separate tab label from visualization title
page6_viz <- create_viz() %>%
  add_viz(
    type = "histogram",
    x_var = "age",
    tabgroup = "demographics",
    title_tabset = "Age",  # Short tab label
   title = "Age Distribution of Survey Respondents by Gender and Region"  # Long viz title
  )

## End(Not run)
```

---

add_vizzes *Add Multiple Visualizations at Once*

Description

Convenience function to add multiple visualizations in a loop by expanding vector parameters. Automatically detects which parameters should be expanded to create multiple visualizations. This is useful when creating many similar visualizations that differ only in one or two parameters.

Usage

```
add_vizzes(
  viz_collection,
  ...,
  .tabgroup_template = NULL,
  .title_template = NULL
)
```

Arguments

`viz_collection`	A viz_collection object from create_viz()
`...`	Visualization parameters. Parameters with multiple values will be expanded to create multiple visualizations. Common parameters with single values will be applied to all visualizations.
`.tabgroup_template`	Optional. Template string for tabgroup with `{i}` placeholder for the iteration index (e.g., `"skills/age/item{i}"`). You can also use parameter names in the template (e.g., `"skills/{y_var}"`). If NULL, tabgroup must be provided as a vector of the same length as expandable parameters.
`.title_template`	Optional. Template string for title with `{i}` placeholder.

Details

The function identifies "expandable" parameters (y_var, x_var, y_var, stack_var, questions) and creates one visualization per value. Other parameters are applied to all visualizations. All expandable vector parameters must have the same length.

Templates use glue syntax:

- `{i}` is replaced with the iteration number (1, 2, 3, ...)
- `{param_name}` is replaced with the current value of that parameter

Value

The updated viz_collection object with multiple visualizations added

Examples

```
## Not run:
# Basic expansion - create 3 timeline visualizations
viz <- create_viz(type = "timeline", time_var = "wave", chart_type = "line") |>
  add_vizzes(
```

```
    y_var = c("SInfo1", "SInfo2", "SInfo3"),
    group_var = "AgeGroup",  # Same for all
    .tabgroup_template = "skills/age/item{i}"
  )

# Parallel expansion - titles match the variables
viz <- create_viz(type = "stackedbar") |>
  add_vizzes(
    x_var = c("Age", "Gender", "Education"),
    title = c("By Age", "By Gender", "By Education"),
    .tabgroup_template = "demographics/demo{i}"
  )

# Use variable names in template
viz <- create_viz(type = "timeline") |>
  add_vizzes(
    y_var = c("SInfo1", "SInfo2", "SInfo3"),
    .tabgroup_template = "skills/{y_var}"
  )

# Helper function pattern
add_all_questions <- function(viz, vars, group_var, tbgrp, demographic, wave) {
  wave_path <- tolower(gsub(" ", "", wave))
  viz |> add_vizzes(
    y_var = vars,
    group_var = group_var,
    .tabgroup_template = glue::glue("{tbgrp}/{wave_path}/{demographic}/item{{i}}")
  )
}

viz <- create_viz(type = "timeline", time_var = "wave") |>
  add_all_questions(
    vars = c("var1", "var2", "var3"),
    group_var = "AgeGroup",
    tbgrp = "skills",
    demographic = "age",
    wave = "Over Time"
  )

## End(Not run)
```

`add_widget` *Add a generic htmlwidget to the dashboard*

Description

Embed any htmlwidget object (plotly, leaflet, echarts4r, DT, etc.) directly into a dashboard page. The widget will be rendered as-is.

Usage

```
add_widget(
  content,
  widget,
  title = NULL,
  height = NULL,
  tabgroup = NULL,
  filter_vars = NULL,
  show_when = NULL
)
```

Arguments

content	A content_collection, page_object, or dashboard_project
widget	An htmlwidget object
title	Optional title displayed above the widget
height	Optional CSS height (e.g., "400px", "50vh")
tabgroup	Optional tabgroup for organizing content
filter_vars	Optional character vector of input filter variables to apply to this block.
show_when	One-sided formula controlling conditional display based on input values.

Value

Updated content object

---

apply_theme *Apply Theme to Dashboard*

Description

Applies a theme to an existing dashboard_project object or returns theme parameters for use in `create_dashboard()`. Supports piping for easy theme application. You can override any theme parameter by passing it as an additional argument.

Usage

```
apply_theme(proj = NULL, theme, ...)
```

Arguments

proj	Optional. A dashboard_project object to apply the theme to. If NULL, returns the theme parameters as a list.
theme	A theme list (e.g., from `theme_ascor()`, `theme_academic()`, etc.)
...	Additional parameters to override theme defaults. Can include any of: `navbar_bg_color`, `navbar_text_color`, `navbar_text_hover_color`, `mainfont`, `fontsize`, `fontcolor`, `linkcolor`, `monofont`, `monobackgroundcolor`, `linestretch`, `backgroundcolor`, `max_width`, `margin_left`, `margin_right`, `margin_top`, `margin_bottom`

Value

If proj is provided, returns the modified dashboard_project object. If proj is NULL, returns the theme list.

Examples

```
## Not run:
# Method 1: Pipe theme directly into dashboard (EASIEST!)
dashboard <- create_dashboard("my_dashboard", "My Research") %>%
  apply_theme(theme_ascor()) %>%
  add_page("Home", text = "# Welcome", is_landing_page = TRUE)

# Method 2: Override specific theme parameters
dashboard <- create_dashboard("tech_dash", "Tech Dashboard") %>%
  apply_theme(theme_modern("purple"), mainfont = "Roboto", fontsize = "18px") %>%
  add_page("Data", visualizations = my_viz)

# Method 3: Get theme parameters only
ascor_params <- apply_theme(theme = theme_ascor())

# Method 4: Customize multiple parameters
dashboard <- create_dashboard("custom", "Custom Dashboard") %>%
  apply_theme(
    theme_clean(),
    mainfont = "Inter",
    fontsize = "18px",
    linkcolor = "#8B0000",
    max_width = "1400px"
  )

## End(Not run)
```

---

ascor_dashboard *Generate an ASCoR-themed dashboard for the University of Amsterdam*

Description

This function creates and renders a professional dashboard with ASCoR (Amsterdam School of Communication Research) and University of Amsterdam branding. The dashboard showcases the dashboardr package with UvA colors, styling, and branding.

Usage

```
ascor_dashboard(directory = "ascor_dashboard")
```

Arguments

directory	Character string. Directory where the dashboard files will be created. Defaults to "ascor_dashboard". Quarto will render HTML to directory/docs/.

Details

The ASCoR dashboard features:

- UvA red (#CB0D0D) as primary branding color
- Professional typography with Inter font
- ASCoR logo in the navbar (if logo file is provided)
- Clean, academic styling appropriate for research communication
- Example visualizations using General Social Survey data

Value

Invisibly returns the dashboard_project object.

Examples

```
## Not run:
# Run the ASCoR dashboard (requires Quarto CLI and 'gssr' package)
ascor_dashboard()

# Specify custom directory
ascor_dashboard(directory = "my_ascor_dashboard")

## End(Not run)
```

---

card *Create a Bootstrap card component*

Description

Helper function to create Bootstrap card components for displaying content in a structured way. Useful for author profiles, feature highlights, or any content that benefits from card layout.

Usage

```
card(
  content,
  title = NULL,
  image = NULL,
  image_alt = NULL,
  footer = NULL,
  class = NULL,
  style = NULL
)
```

Arguments

content	Card content (text, HTML, or other elements)
title	Optional card title
image	Optional image URL or path
image_alt	Alt text for the image
footer	Optional card footer content
class	Additional CSS classes for the card
style	Additional inline styles for the card

Value

HTML div element with Bootstrap card classes

Examples

```
## Not run:
# Simple text card
card("This is a simple card with just text content")

# Card with title and image
card(
  content = "This is the card body content",
  title = "Card Title",
  image = "https://example.com/image.jpg",
  image_alt = "Description of image"
)

# Author card
card(
 content = "Dr. Jane Smith is a researcher specializing in data science and visualization.",
  title = "Dr. Jane Smith",
  image = "https://example.com/jane.jpg",
  footer = "Website: janesmith.com"
)

## End(Not run)
```

---

card_row *Display cards in a Bootstrap row*

Description

Helper function to display multiple cards in a responsive Bootstrap row layout.

Usage

```
card_row(..., cols = 2, class = NULL)
```

Arguments

...	Card objects to display
cols	Number of columns per row (default: 2)
class	Additional CSS classes for the row

Value

HTML div element with Bootstrap row classes containing the cards

Examples

```
## Not run:
# Display two cards in a row
card_row(card1, card2)

# Display three cards in a row (3 columns)
card_row(card1, card2, card3, cols = 3)

## End(Not run)
```

---

combine_content *Combine content collections (universal combiner)*

Description

Universal function to combine content_collection or viz_collection objects. Preserves all content types (visualizations, pagination markers, text blocks) and collection-level attributes (lazy loading, etc.).

Usage

```
combine_content(...)
```

Arguments

...	One or more content_collection or viz_collection objects

Value

Combined content_collection

Examples

```
## Not run:
# Combine multiple collections
all_viz <- demo_viz %>%
  combine_content(analysis_viz) %>%
  combine_content(summary_viz)

# With pagination
paginated <- section1_viz %>%
  combine_content(section2_viz) %>%
  add_pagination() %>%
  combine_content(section3_viz)

# Using + operator
combined <- viz1 + viz2 + viz3

## End(Not run)
```

`combine_viz` *Combine visualization collections*

Description

This function has been superseded by `combine_content()`. It still works but we recommend using `combine_content()` for new code as it handles all content types and attributes more reliably.

Usage

```
combine_viz(...)
```

Arguments

`...`	One or more viz_collection objects to combine

Value

A combined viz_collection

Examples

```
## Not run:
viz1 <- create_viz() %>% add_viz(type = "histogram", x_var = "age")
viz2 <- create_viz() %>% add_viz(type = "histogram", x_var = "income")
combined <- combine_viz(viz1, viz2)  # Combines both

## End(Not run)
```

create_blockquote *Create a Styled Blockquote*

Description

Creates a custom-styled blockquote with customizable colors, borders, and styling. Useful for highlighting questions, quotes, or important text in dashboards.

Usage

```
create_blockquote(
  text,
  preset = NULL,
  class_name = "custom-blockquote",
  font_size = "1em",
  text_color = "#333",
  border_width = "5px",
  border_color = "#0056b3",
  background_color = "#f0f8ff",
  padding = "10px 20px",
  margin = "20px 0",
  line_height = "1.6",
  return_css = FALSE,
  use_class = FALSE
)
```

Arguments

text	Character string. The text content to display in the blockquote.
preset	Either a character string for built-in presets ("question", "info", "warning", "success", "error", "note") OR a named list with custom styling parameters (e.g., list(border_color = "#0056b3", background_color = "#e3f2fd")). Default is NULL (uses default styling).
class_name	Character string. CSS class name for the blockquote. Default is "custom-blockquote".
font_size	Character string. Font size (e.g., "1em", "16px"). Default is "1em".
text_color	Character string. Text color (hex, rgb, or named color). Default is "#333".
border_width	Character string. Left border width (e.g., "5px", "3px"). Default is "5px".
border_color	Character string. Left border color. Default is "#0056b3".
background_color	Character string. Background color. Default is "#f0f8ff".
padding	Character string. Padding inside the blockquote. Default is "10px 20px".
margin	Character string. Margin around the blockquote. Default is "20px 0".
line_height	Character string. Line height for text. Default is "1.6".

return_css	Logical. If TRUE, returns only the CSS. If FALSE (default), returns HTML with inline CSS.
use_class	Logical. If TRUE, returns HTML with class reference and separate CSS block. If FALSE (default), uses inline styles.

Value

If use_class = FALSE: HTML blockquote with inline styles. If use_class = TRUE: List with html and css elements. If return_css = TRUE: Only the CSS string.

Examples

```
# Basic usage with defaults
create_blockquote("This is an important question about data quality.")

# Using built-in presets (as strings)
create_blockquote("How do you rate our service?", preset = "question")
create_blockquote("Please check your input.", preset = "warning")
create_blockquote("Operation completed!", preset = "success")

# Using custom presets (as lists) - pass directly!
algosoc_style <- list(
  border_color = "#0056b3",
  background_color = "#e3f2fd",
  text_color = "#1565c0"
)
create_blockquote("AlgoSoc question here", preset = algosoc_style)

# Define multiple custom styles and reuse
survey_style <- list(border_color = "#6f42c1", background_color = "#f8f5ff")
important_style <- list(
  border_color = "#e74c3c",
  background_color = "#ffebee",
  border_width = "8px"
)

create_blockquote("Survey question 1", preset = survey_style)
create_blockquote("Survey question 2", preset = survey_style)
create_blockquote("IMPORTANT!", preset = important_style)

# Custom styling (overriding preset)
create_blockquote(
  "Warning: Please review the data before proceeding.",
  preset = "warning",
  border_width = "8px",  # Override preset border width
  font_size = "1.2em"     # Override preset font size
)

# Fully custom (no preset)
create_blockquote(
  "How satisfied are you with our service?",
  border_color = "#6f42c1",
```

```
  background_color = "#f8f5ff",
  font_size = "1.1em",
  padding = "15px 25px"
)

# Using class-based approach (for multiple blockquotes)
result <- create_blockquote(
  "Question 1: What is your opinion?",
  preset = "question",
  use_class = TRUE
)
# View the CSS component (an htmltools tag)
result$css
# View the HTML component
result$html
```

`create_content` *Create a new content/visualization collection (alias for create_viz)*

Description

This is an alias for `create_viz` - both functions are identical. Use whichever name makes more sense for your use case. The returned collection can be built up with any combination of add_viz(), add_text(), and add_image().

Usage

```
create_content(
  data = NULL,
  tabgroup_labels = NULL,
  shared_first_level = TRUE,
  ...
)
```

Arguments

`data`	Optional data frame to use for all visualizations in this collection. This data will be used by add_viz() calls and can be used with preview().
`tabgroup_labels`	Named vector/list mapping tabgroup IDs to display names
`shared_first_level`	Logical. When TRUE (default), multiple first-level tabgroups will share a single tabset. When FALSE, each first-level tabgroup is rendered as a separate section (stacked vertically).
`...`	Default parameters to apply to all subsequent add_viz() calls. Common defaults include: type, color_palette, stacked_type, horizontal, etc. Any parameter that can be passed to add_viz() can be set as a default here.

Details

Note: Both names return the same object with both "content_collection" and "viz_collection" classes for backward compatibility.

Value

A content_collection (also a viz_collection for compatibility)

Examples

```
## Not run:
# Create content with inline data for preview
content <- create_content(data = mtcars) %>%
  add_text("# MPG Analysis") %>%
  add_viz(type = "histogram", x_var = "mpg") %>%
  preview()

# Set shared defaults like type - all add_viz() calls inherit these
content <- create_content(
  data = survey_df,
  type = "stackedbar",
  stacked_type = "percent",
  horizontal = TRUE
) %>%
  add_viz(x_var = "age", stack_var = "response", tabgroup = "Age") %>%
  add_viz(x_var = "gender", stack_var = "response", tabgroup = "Gender")

# These are equivalent:
content <- create_content() %>%
  add_text("# Title") %>%
  add_viz(type = "histogram", x_var = "age")

content <- create_viz() %>%
  add_text("# Title") %>%
  add_viz(type = "histogram", x_var = "age")

## End(Not run)
```

---

create_dashboard *Create a new dashboard project*

Description

Initializes a dashboard project object that can be built up using the piping workflow with add_landingpage() and add_page().

Usage

```
create_dashboard(
  output_dir = "site",
  title = "Dashboard",
  logo = NULL,
  favicon = NULL,
  github = NULL,
  twitter = NULL,
  linkedin = NULL,
  email = NULL,
  website = NULL,
  search = TRUE,
  theme = NULL,
  custom_css = NULL,
  custom_scss = NULL,
  tabset_theme = "minimal",
  tabset_colors = NULL,
  author = NULL,
  description = NULL,
  page_footer = NULL,
  date = NULL,
  sidebar = FALSE,
  sidebar_style = "docked",
  sidebar_background = "light",
  sidebar_foreground = NULL,
  sidebar_border = TRUE,
  sidebar_alignment = "left",
  sidebar_collapse_level = 2,
  sidebar_pinned = FALSE,
  sidebar_tools = NULL,
  sidebar_contents = NULL,
  breadcrumbs = TRUE,
  page_navigation = FALSE,
  back_to_top = FALSE,
  reader_mode = FALSE,
  repo_url = NULL,
  repo_actions = NULL,
  navbar_style = NULL,
  navbar_bg_color = NULL,
  navbar_text_color = NULL,
  navbar_text_hover_color = NULL,
  navbar_brand = NULL,
  navbar_toggle = NULL,
  max_width = NULL,
  mainfont = "Fira Sans",
  fontsize = "16px",
  fontcolor = NULL,
  linkcolor = NULL,
```

```
    monofont = "Fira Code",
    monobackgroundcolor = NULL,
    linestretch = NULL,
    backgroundcolor = NULL,
    margin_left = NULL,
    margin_right = NULL,
    margin_top = NULL,
    margin_bottom = NULL,
    math = NULL,
    code_folding = NULL,
    code_tools = NULL,
    toc = NULL,
    toc_depth = 3,
    google_analytics = NULL,
    plausible = NULL,
    gtag = NULL,
    value_boxes = FALSE,
    metrics_style = NULL,
    page_layout = "full",
    mobile_toc = FALSE,
    viewport_width = NULL,
    viewport_scale = NULL,
    viewport_user_scalable = TRUE,
    self_contained = FALSE,
    code_overflow = NULL,
    html_math_method = NULL,
    shiny = FALSE,
    observable = FALSE,
    jupyter = FALSE,
    publish_dir = NULL,
    github_pages = NULL,
    netlify = NULL,
    allow_inside_pkg = FALSE,
    warn_before_overwrite = TRUE,
    sidebar_groups = NULL,
    navbar_sections = NULL,
    lazy_load_charts = FALSE,
    lazy_load_margin = "200px",
    lazy_load_tabs = NULL,
    lazy_debug = FALSE,
    pagination_separator = "of",
    pagination_position = "bottom",
    powered_by_dashboardr = TRUE,
    chart_export = FALSE,
    backend = "highcharter",
    contextual_viz_errors = FALSE,
    url_params = FALSE,
    cross_tab_data_mode = c("inline", "asset"),
```

```
  min_cell_size = 0L,
  rds_bundle_threshold = 5L,
  deferred_charts = FALSE
)
```

Arguments

output_dir	Directory for generated files
title	Overall title for the dashboard site
logo	Optional logo filename (will be copied to output directory)
favicon	Optional favicon filename (will be copied to output directory)
github	GitHub repository URL (optional)
twitter	Twitter profile URL (optional)
linkedin	LinkedIn profile URL (optional)
email	Email address (optional)
website	Website URL (optional)
search	Enable search functionality (default: TRUE)
theme	Bootstrap theme (cosmo, flatly, journal, etc.) (optional)
custom_css	Path to custom CSS file (optional)
custom_scss	Path to custom SCSS file (optional)
tabset_theme	Tabset theme: "minimal" (default), "modern", "pills", "classic", "underline", "segmented", or "none"
tabset_colors	Named list of tabset colors (e.g., list(active_bg = "#2563eb"))
author	Author name for the site (optional)
description	Site description for SEO (optional)
page_footer	Custom footer text (optional)
date	Site creation/update date (optional)
sidebar	Enable/disable global sidebar (default: FALSE)
sidebar_style	Sidebar style (floating, docked, etc.) (optional)
sidebar_background	Sidebar background color (optional)
sidebar_foreground	Sidebar foreground (text) color (optional)
sidebar_border	Whether to show sidebar border (default TRUE)
sidebar_alignment	Sidebar alignment: "left" (default) or "right"
sidebar_collapse_level	Heading level at which sidebar items collapse (default 2)
sidebar_pinned	Whether sidebar is pinned open (default FALSE)
sidebar_tools	Sidebar tools configuration (optional)

`sidebar_contents`
: Sidebar contents configuration (optional)

`breadcrumbs`
: Whether to show breadcrumbs navigation (default TRUE)

`page_navigation`
: Whether to show prev/next page navigation (default FALSE)

`back_to_top`
: Whether to show a back-to-top button (default FALSE)

`reader_mode`
: Whether to enable reader mode (default FALSE)

`repo_url`
: Repository URL for source code link (optional)

`repo_actions`
: Repository actions configuration (optional)

`navbar_style`
: Navbar style (default, dark, light) (optional)

`navbar_bg_color`
: Navbar background color (CSS color value, e.g., "#2563eb", "rgb(37, 99, 235)") (optional)

`navbar_text_color`
: Navbar text color (CSS color value, e.g., "#ffffff", "rgb(255, 255, 255)") (optional)

`navbar_text_hover_color`
: Navbar text color on hover (CSS color value, e.g., "#f0f0f0") (optional)

`navbar_brand`
: Custom brand text (optional)

`navbar_toggle`
: Mobile menu toggle behavior (optional)

`max_width`
: Maximum width for page content (e.g., "1400px", "90%") (optional)

`mainfont`
: Font family for document text. Recommended: "Fira Sans" (smooth, modern), "Lato" (warm), "Source Sans Pro" (elegant), or "Roboto" (technical). Default is "Fira Sans" for a smooth, professional look.

`fontsize`
: Base font size for document (default: "16px" for optimal readability)

`fontcolor`
: Default text color (e.g., "#1f2937" for readable dark gray) (optional)

`linkcolor`
: Default hyperlink color (e.g., "#2563eb" for vibrant blue) (optional)

`monofont`
: Font family for code elements. Recommended: "Fira Code" (with ligatures), "JetBrains Mono", "Source Code Pro", or "IBM Plex Mono". Default: "Fira Code".

`monobackgroundcolor`
: Background color for code elements (e.g., "#f8fafc" for subtle gray) (optional)

`linestretch`
: Line height for text (default: 1.5) (optional)

`backgroundcolor`
: Background color for document (optional)

`margin_left`
: Left margin for document body (optional)

`margin_right`
: Right margin for document body (optional)

`margin_top`
: Top margin for document body (optional)

`margin_bottom`
: Bottom margin for document body (optional)

`math`
: Enable/disable math rendering (katex, mathjax) (optional)

`code_folding`
: Code folding behavior (none, show, hide) (optional)

code_tools	Code tools (copy, download, etc.) (optional)
toc	Table of contents (floating, left, right) (optional)
toc_depth	TOC depth level (default: 3)
google_analytics	Google Analytics ID (optional)
plausible	Plausible analytics script hash (e.g., "pa-UnPiJwxFi8TS"). Find your script hash in Plausible Settings > Tracking Code (Script Installation tab). This format includes ad-blocker bypass and doesn't require specifying your domain.
gtag	Google Tag Manager ID (optional)
value_boxes	Enable value box styling (default: FALSE)
metrics_style	Metrics display style (optional)
page_layout	Quarto page layout mode. Default is "full" for better mobile responsiveness. Other options: "article" (constrained width), "custom". See Quarto docs for details.
mobile_toc	Logical. If TRUE, adds a collapsible mobile-friendly TOC button that appears in the top-right corner. Useful for mobile/tablet viewing. Default: FALSE.
viewport_width	Numeric or character. Controls mobile viewport behavior. Default is NULL (standard responsive behavior). Set to a number (e.g., 1200) to force desktop rendering width on mobile devices. Useful if charts look squished on mobile. Can also be a full viewport string like "width=1400, minimum-scale=0.5" for advanced control.
viewport_scale	Numeric. Initial zoom scale for mobile devices (e.g., 0.3 to zoom out, 1.0 for no zoom). Only used if viewport_width is set. Default: NULL (no scale specified).
viewport_user_scalable	Logical. Allow users to pinch-zoom on mobile? Default: TRUE. Only relevant if viewport_width is set.
self_contained	Logical. If TRUE, produces a standalone HTML file with all dependencies embedded. Makes files larger but more portable and can improve mobile rendering consistency. Default: FALSE.
code_overflow	Character. Controls code block overflow behavior. Options: "wrap" (wrap long lines), "scroll" (horizontal scrollbar). Default: NULL (Quarto default). Set to "wrap" to prevent horizontal scrolling issues on mobile.
html_math_method	Character. Method for rendering math equations. Options: "mathjax", "katex", "webtex", "gladtex", "mathml". Default: NULL (Quarto default).
shiny	Enable Shiny interactivity (default: FALSE)
observable	Enable Observable JS (default: FALSE)
jupyter	Enable Jupyter widgets (default: FALSE)
publish_dir	Custom publish directory (optional)
github_pages	GitHub Pages configuration (optional)
netlify	Netlify deployment settings (optional)
allow_inside_pkg	Allow output directory inside package (default FALSE)

warn_before_overwrite
: Warn before overwriting existing files (default TRUE)

sidebar_groups
: List of sidebar groups for hybrid navigation (optional)

navbar_sections
: List of navbar sections that link to sidebar groups (optional)

lazy_load_charts
: Enable lazy loading for charts (default: FALSE). When TRUE, charts only render when they scroll into view, dramatically improving initial page load time for pages with many visualizations.

lazy_load_margin
: Distance from viewport to start loading charts (default: "200px"). Larger values mean charts start loading earlier.

lazy_load_tabs
: Only render charts in the active tab (default: TRUE when lazy_load_charts is TRUE). Charts in hidden tabs load when the tab is clicked.

lazy_debug
: Enable debug logging to browser console for lazy loading (default: FALSE). When TRUE, prints timing information for each chart load.

pagination_separator
: Text to show in pagination navigation (e.g., "of" -> "1 of 3"), default: "of". Applies to all paginated pages unless overridden at page level.

pagination_position
: Default position for pagination controls: "bottom" (default, sticky at bottom), "top" (inline with page title), or "both" (top and bottom). This sets the default for all paginated pages. Individual pages can override this by passing position to add_pagination().

powered_by_dashboardr
: Whether to automatically add "Powered by dashboardr" branding (default: TRUE). When TRUE, adds a badge-style branding element. Can be overridden by explicitly calling add_powered_by_dashboardr() with custom options, or set to FALSE to disable entirely.

chart_export
: Whether to enable chart export functionality (default FALSE)

backend
: Rendering backend: "highcharter" (default), "plotly", "echarts4r", or "ggiraph".

contextual_viz_errors
: Logical. If TRUE, generated visualization chunks wrap viz calls in tryCatch and prepend contextual labels (title/type) to error messages. Default: FALSE.

url_params
: Logical. If TRUE, enable URL parameter support for inputs. Default: FALSE.

cross_tab_data_mode
: How cross-tab data is embedded: "inline" (default, current behavior) or "asset" (write to external .json files for lazy loading). Asset mode reduces HTML size dramatically for dashboards with many filtered charts.

min_cell_size
: Integer. Minimum cell count for privacy protection in cross-tab data. Rows where 0 < n < min_cell_size are suppressed (removed) from cross-tab output. Set to 0 to disable suppression. Default: 0 (opt-in).

rds_bundle_threshold
: Integer. If the dashboard has at least this many pending generated datasets, write them as one bundled `.rds` list file instead of many individual `.rds` files. Default: 5. Set to 0 or Inf to disable bundling.

deferred_charts
: Logical. If TRUE, only render initially visible charts; others become lightweight placeholders that load on demand when revealed by show_when. Default: FALSE.

Value

A dashboard_project object

Examples

```
## Not run:
# Basic dashboard
dashboard <- create_dashboard("my_dashboard", "My Analysis Dashboard")

# Comprehensive dashboard with all features
dashboard <- create_dashboard(
  "my_dashboard",
  "My Analysis Dashboard",
  logo = "logo.png",
  github = "https://github.com/username/repo",
  twitter = "https://twitter.com/username",
  theme = "cosmo",
  author = "Dr. Jane Smith",
  description = "Comprehensive data analysis dashboard",
  page_footer = "(c) 2024 Company Name",
  sidebar = TRUE,
  toc = "floating",
  google_analytics = "GA-XXXXXXXXX",
  value_boxes = TRUE,
  shiny = TRUE
)

# Dashboard with lazy loading for better performance
dashboard <- create_dashboard(
  "fast_dashboard",
  "High Performance Dashboard",
  lazy_load_charts = TRUE,
  lazy_load_margin = "300px",
  lazy_load_tabs = TRUE
)

# Professional styling with modern fonts (Google Fonts work great!)
dashboard <- create_dashboard(
  "styled_dashboard",
  "Beautifully Styled Dashboard",
  navbar_bg_color = "#1e40af",     # Deep blue navbar
  mainfont = "Fira Sans",           # Smooth, modern (default choice)
  fontsize = "16px",
  fontcolor = "#1f2937",            # Dark gray for readability
  linkcolor = "#2563eb",            # Vibrant blue links
  monofont = "Fira Code",           # Code font with ligatures
  monobackgroundcolor = "#f8fafc",  # Light gray code background
  linestretch = 1.6,                # Comfortable line spacing
```

```
  backgroundcolor = "#ffffff"
)

# Alternative professional font combinations:
# Option 1: Warm & Friendly
dashboard <- create_dashboard(
  "friendly_dashboard",
  title = "Friendly Dashboard",
  mainfont = "Lato",               # Warm, approachable
  monofont = "JetBrains Mono"      # Excellent for code
)

# Option 2: Elegant & Refined
dashboard <- create_dashboard(
  "elegant_dashboard",
  title = "Elegant Dashboard",
  mainfont = "Source Sans Pro",    # Elegant, highly readable
  monofont = "Source Code Pro"     # Matching code font
)

# Option 3: Technical Feel
dashboard <- create_dashboard(
  "tech_dashboard",
  title = "Tech Dashboard",
  mainfont = "Roboto",             # Technical, clean
  monofont = "JetBrains Mono"      # Excellent for code
)

## End(Not run)
```

`create_loading_overlay` *Create a loading overlay for a dashboard page*

Description

Creates an animated loading overlay that appears when the page loads and automatically fades out after a specified duration. Useful for providing visual feedback while charts and visualizations are rendering.

Usage

```
create_loading_overlay(
  text = "Loading",
  timeout_ms = 2200,
  theme = c("light", "glass", "dark", "accent")
)
```

Arguments

text	Text to display in the loading overlay (default: "Loading")
timeout_ms	Duration in milliseconds before the overlay hides (default: 2200)
theme	Visual theme for the overlay. One of: • "light" - Clean white overlay with subtle shadow • "glass" - Glassmorphic semi-transparent overlay • "dark" - Dark gradient overlay • "accent" - Light overlay with blue accents

Value

An htmltools tag object containing the overlay HTML, CSS, and JavaScript

Examples

```
## Not run:
# In a Quarto document R chunk:
dashboardr::create_loading_overlay("Loading Dashboard...", 2000, "glass")

## End(Not run)
```

---

create_page *Create a page object*

Description

Creates a standalone page object that can be populated with content and later added to a dashboard. Pages can have visualizations added directly (without creating separate content objects), making this the simplest way to build dashboards.

Usage

```
create_page(
  name,
  data = NULL,
  data_path = NULL,
  type = NULL,
  color_palette = NULL,
  icon = NULL,
  is_landing_page = FALSE,
  navbar_align = c("left", "right"),
  tabset_theme = NULL,
  tabset_colors = NULL,
  overlay = FALSE,
  overlay_theme = c("light", "glass", "dark", "accent"),
```

```
  overlay_text = "Loading",
  overlay_duration = 2200,
  lazy_load_charts = NULL,
  lazy_load_margin = NULL,
  lazy_load_tabs = NULL,
  lazy_debug = NULL,
  pagination_separator = NULL,
  time_var = NULL,
  weight_var = NULL,
  filter = NULL,
  drop_na_vars = NULL,
  shared_first_level = TRUE,
  ...
)
```

Arguments

name	Page display name (required)
data	Data frame for this page. All visualizations on this page will automatically use this data (no need to specify data separately).
data_path	Path to existing data file (alternative to data parameter)
type	Default visualization type for add_viz() calls (e.g., "bar", "histogram", "stacked-bar")
color_palette	Default color palette for all visualizations on this page
icon	Optional iconify icon shortcode (e.g., "ph:users-three", "ph:chart-line")
is_landing_page	Whether this should be the landing page (default: FALSE)
navbar_align	Position of page in navbar: "left" (default) or "right"
tabset_theme	Optional tabset theme for this page
tabset_colors	Optional tabset colors for this page
overlay	Whether to show a loading overlay on page load (default: FALSE)
overlay_theme	Theme for loading overlay: "light", "glass", "dark", or "accent"
overlay_text	Text to display in loading overlay (default: "Loading")
overlay_duration	Duration in milliseconds for overlay (default: 2200)
lazy_load_charts	Override dashboard-level lazy loading for this page
lazy_load_margin	Override viewport margin for lazy loading
lazy_load_tabs	Override tab-aware lazy loading for this page
lazy_debug	Override debug mode for lazy loading
pagination_separator	Text for pagination navigation (e.g., "of" -> "1 of 3")

time_var	Name of the time/x-axis column for input filters
weight_var	Name of weight variable for weighted visualizations (applies to all viz)
filter	Filter expression for subsetting data (e.g., ~ year >= 2020)
drop_na_vars	Default for dropping NA values in visualizations
shared_first_level	Logical. When TRUE (default), multiple first-level tabgroups will share a single tabset. When FALSE, each first-level tabgroup is rendered as a separate section (stacked vertically).
...	Additional default parameters passed to all add_viz() calls

Value

A page_object that can be modified with add_viz(), add_text(), etc.

Examples

```
## Not run:
# SIMPLE: Add visualizations directly to the page!
# No need to create separate content objects
analysis <- create_page("Analysis", data = gss, type = "bar") %>%
  add_viz(x_var = "degree", title = "Education") %>%
  add_viz(x_var = "race", title = "Race") %>%
  add_viz(x_var = "happy", title = "Happiness", type = "stackedbar", stack_var = "sex")

# LANDING PAGE: Just add text
home <- create_page("Home", icon = "ph:house-fill", is_landing_page = TRUE) %>%
  add_text("# Welcome!", "", "Explore our data dashboard.") %>%
  add_callout("Data updated weekly", type = "tip")

# MIXED: Combine direct viz with pre-built content
trends <- create_page("Trends", data = gss) %>%
  add_viz(x_var = "year", y_var = "happy", type = "timeline") %>%
  add_content(detailed_analysis)  # Add pre-built content too

# PREVIEW: See what the page looks like before adding to dashboard
analysis %>% preview()

# BUILD DASHBOARD
create_dashboard(title = "My Dashboard") %>%
  add_pages(home, analysis, trends) %>%
  generate_dashboard()

## End(Not run)
```

create_pagination_nav *Create pagination navigation controls for a dashboard page*

Description

Creates navigation controls for multi-page dashboards with Previous/Next buttons and page indicator. Use this in your QMD file to add clean pagination without embedding HTML directly.

Usage

```
create_pagination_nav(page_num, total_pages, base_name, position = "top")
```

Arguments

page_num	Current page number
total_pages	Total number of pages
base_name	Base filename (e.g., "knowledge" for knowledge.qmd, knowledge_p2.qmd, etc.)
position	Position of navigation: "top", "bottom", or "both" (default: "top")

Value

An htmltools tag object containing the pagination HTML and JavaScript

Examples

```
## Not run:
# In a Quarto document R chunk with results='asis':
dashboardr::create_pagination_nav(1, 3, "knowledge", "top")

# For both top and bottom:
dashboardr::create_pagination_nav(1, 3, "knowledge", "both")

## End(Not run)
```

create_viz *Create a new visualization collection*

Description

Initializes an empty collection for building up multiple visualizations using the piping workflow. Optionally accepts custom display labels for tab groups and default parameters that apply to all visualizations.

Usage

```
create_viz(data = NULL, tabgroup_labels = NULL, shared_first_level = TRUE, ...)
```

Arguments

data	Optional data frame to use for all visualizations in this collection. This data will be used by add_viz() calls and can be used with preview(). Can also be passed to add_page() which will use this as fallback if no page-level data is provided.
tabgroup_labels	Named vector/list mapping tabgroup IDs to display names
shared_first_level	Logical. When TRUE (default), multiple first-level tabgroups will share a single tabset. When FALSE, each first-level tabgroup is rendered as a separate section (stacked vertically).
...	Default parameters to apply to all subsequent add_viz() calls. Any parameter specified in add_viz() will override the default. Useful for setting common parameters like type, color_palette, stacked_type, etc.

Value

A viz_collection object

Examples

```
## Not run:
# Create viz collection with data for preview
vizzes <- create_viz(data = mtcars) %>%
  add_viz(type = "histogram", x_var = "mpg", title = "MPG Distribution") %>%
  preview()

# Create viz collection with custom group labels
vizzes <- create_viz(tabgroup_labels = c("demo" = "Demographics",
                                         "pol" = "Political Views"))

# Create viz collection with shared defaults
vizzes <- create_viz(
  type = "stackedbars",
  stacked_type = "percent",
  color_palette = c("#d7191c", "#fdae61", "#2b83ba"),
  horizontal = TRUE,
  x_label = ""
) %>%
  add_viz(title = "Wave 1", filter = ~ wave == 1) %>%  # Uses defaults
 add_viz(title = "Wave 2", filter = ~ wave == 2, horizontal = FALSE)  # Overrides horizontal

## End(Not run)
```

`dashboardr_mcp_server` *Start dashboardr MCP Server*

Description

Launches an MCP server that exposes dashboardr documentation, function reference, example code, and visualization guides to LLM-powered coding assistants like Claude Desktop, Claude Code, Cursor, and VS Code Copilot.

Usage

```
dashboardr_mcp_server(
  backend = c("mcptools", "mcpr"),
  transport = c("stdio", "http"),
  port = 8080L
)
```

Arguments

`backend`	Character, which MCP backend to use. `"mcptools"` (default, Posit-maintained) or `"mcpr"` (alternative). If the chosen backend is not installed, falls back to the other.
`transport`	Character, either `"stdio"` (default) or `"http"`.
`port`	Integer, port for HTTP transport (default 8080). Ignored when `transport = "stdio"`.

Details

Tools are defined using `ellmer::tool()` and served via `mcptools::mcp_server()` (default) or `mcpr::serve_io()` as a fallback.

The server exposes five tools:

dashboardr_guide Returns the full dashboardr API guide covering the three-layer architecture, function index, and quick start.

dashboardr_function_help Look up detailed help for any exported dashboardr function.

dashboardr_list_functions List exported functions, optionally filtered by category.

dashboardr_example Get runnable example code for common dashboard patterns.

dashboardr_viz_types Quick reference of all visualization types with key parameters and use cases.

Value

Called for its side effect (starts a blocking MCP server). Does not return.

Configuration

Claude Code:

```
claude mcp add dashboardr -- Rscript -e "dashboardr::dashboardr_mcp_server()"
```

Claude Desktop (`claude_desktop_config.json`):

```
{
  "mcpServers": {
    "dashboardr": {
      "command": "Rscript",
      "args": ["-e", "dashboardr::dashboardr_mcp_server()"]
    }
  }
}
```

VS Code / Cursor:

```
{
  "mcp": {
    "servers": {
      "dashboardr": {
        "type": "stdio",
        "command": "Rscript",
        "args": ["-e", "dashboardr::dashboardr_mcp_server()"]
      }
    }
  }
}
```

`enable_accessibility` *Enable Accessibility Enhancements*

Description

Adds CSS and JavaScript for WCAG 2.1 AA accessibility improvements: focus indicators, modal focus trapping, tab keyboard navigation, ARIA live region for filter announcements, and reduced motion support.

Usage

```
enable_accessibility()
```

Details

Called automatically during page generation.

Value

HTML tags to include accessibility enhancements

---

enable_chart_export *Enable chart export buttons (PNG/SVG/PDF/CSV)*

Description

Injects a script that enables Highcharts export functionality on all charts. Charts will display a hamburger menu button that allows downloading in various formats (PNG, SVG, PDF, CSV, full-screen view).

Usage

```
enable_chart_export()
```

Details

This is typically called automatically when `chart_export = TRUE` is set in `create_dashboard()`.

Value

HTML script tag that enables chart exporting

---

enable_inputs *Enable Input Filter Functionality*

Description

Adds input filter CSS and JavaScript to enable interactive filtering of Highcharts visualizations via multi-select dropdowns. Uses Choices.js for a polished multi-select experience.

Usage

```
enable_inputs(linked = FALSE, show_when = FALSE, url_params = FALSE)
```

Arguments

`linked`	If TRUE, also include script for linked (cascading) parent-child select inputs. Set automatically when the page uses `add_linked_inputs()`.
`show_when`	If TRUE, also include script for conditional viz visibility (`show_when` in `add_viz()`). Set automatically when the page uses it.
`url_params`	If TRUE, also include script for URL parameter support.

Value

HTML tags to include input filter functionality

Examples

```
## Not run:
# In your dashboard page content:
enable_inputs()
enable_inputs(linked = TRUE)  # when using add_linked_inputs()
enable_inputs(show_when = TRUE)  # when using show_when in add_viz()

## End(Not run)
```

---

enable_modals *Enable Modal Functionality*

Description

Adds modal CSS and JavaScript to enable clickable links that open content in a centered modal overlay instead of navigating to a new page.

Usage

```
enable_modals()
```

Value

HTML tags to include modal functionality

Examples

```
## Not run:
# In your dashboard page content:
enable_modals()

## End(Not run)
```

enable_show_when *Enable show_when (conditional visibility) script only*

Description

Includes the JavaScript that evaluates `data-show-when` on viz containers. Called automatically when a page has visualizations with `show_when` but no inputs.

Usage

```
enable_show_when()
```

Value

HTML script tag for show_when.js

enable_sidebar *Enable Sidebar Styling*

Description

Adds modern CSS styling for sidebar panels. Called automatically when a page includes a sidebar via add_sidebar().

Usage

```
enable_sidebar()
```

Value

HTML tags to include sidebar CSS

Examples

```
## Not run:
# Usually called automatically, but can be added manually:
enable_sidebar()

## End(Not run)
```

enable_url_params *Enable URL Parameter Deep Linking*

Description

Adds JavaScript that reads filter state from URL query parameters on page load and updates the URL as filters change. Enables shareable dashboard URLs with pre-set filters and tab navigation via hash fragments.

Usage

```
enable_url_params()
```

Value

HTML script tag for url_params.js

end_input_row *End an input row*

Description

Closes an input row and returns to the parent content collection. Must be called after add_input_row() and all add_input() calls.

Usage

```
end_input_row(row_container)
```

Arguments

row_container	Input row container object

Value

The parent content_collection for further piping

Examples

```
## Not run:
content <- create_content() %>%
  add_input_row() %>%
    add_input(input_id = "filter1", filter_var = "var1", options = c("A", "B")) %>%
    add_input(input_id = "filter2", filter_var = "var2", options = c("X", "Y")) %>%
  end_input_row() %>%
  add_text("Content after the input row...")

## End(Not run)
```

end_layout_column *End a manual layout column*

Description

End a manual layout column

Usage

```
end_layout_column(column_container)
```

Arguments

column_container	A layout_column_container created by `add_layout_column()`.

Value

The parent content_collection or page_object.

end_layout_row *End a manual layout row*

Description

End a manual layout row

Usage

```
end_layout_row(row_container)
```

Arguments

row_container	A layout_row_container created by `add_layout_row()`.

Value

The parent layout_column_container.

end_sidebar *End a sidebar*

Description

Closes a sidebar container and returns to the parent content collection. Must be called after add_sidebar() and all content additions.

Usage

```
end_sidebar(sidebar_container)
```

Arguments

sidebar_container	Sidebar container object created by add_sidebar()

Value

The parent content_collection or page_object for further piping

Examples

```
## Not run:
content <- create_content() %>%
  add_sidebar() %>%
    add_text("## Filters") %>%
    add_input(input_id = "filter1", filter_var = "var1", options = c("A", "B")) %>%
  end_sidebar() %>%
  add_text("Content after the sidebar...")

## End(Not run)
```

end_value_box_row *End a value box row*

Description

Closes a value box row and returns to the parent content collection. Must be called after add_value_box_row() and all add_value_box() calls.

Usage

```
end_value_box_row(row_container)
```

Arguments

row_container	Value box row container object

Examples

```
## Not run:
content <- create_content() %>%
  add_value_box_row() %>%
    add_value_box(title = "Users", value = "1,234") %>%
    add_value_box(title = "Revenue", value = "EUR 56K") %>%
  end_value_box_row() %>%
  add_text("More content after the row...")

## End(Not run)
```

---

generate_dashboard *Generate all dashboard files*

Description

Writes out all .qmd files, _quarto.yml, and optionally renders the dashboard to HTML using Quarto. Supports incremental builds to skip unchanged pages and preview mode to generate only specific pages.

Usage

```
generate_dashboard(
  proj,
  render = TRUE,
  open = "browser",
  incremental = FALSE,
  preview = NULL,
  show_progress = TRUE,
  quiet = FALSE,
  standalone = FALSE
)
```

Arguments

proj	A dashboard_project object
render	Whether to render to HTML (requires Quarto CLI)
open	How to open the result: "browser", "viewer", or FALSE
incremental	Whether to use incremental builds (default: FALSE). When TRUE, skips regenerating QMD files for unchanged pages and skips Quarto rendering if nothing changed. Uses MD5 hashing to detect changes.
preview	Optional character vector of page names to generate. When specified, only the listed pages will be generated, skipping all others. Useful for quick testing of specific pages without waiting for the entire dashboard to generate. Page names are case-insensitive. If a page name doesn't exist, the function will suggest alternatives based on typo detection. Default: NULL (generates all pages).

show_progress Whether to display custom progress indicators (default: TRUE). When TRUE, shows a beautiful progress display with timing information, progress bars, and visual indicators for each generation stage. Set to FALSE for minimal output.

quiet Whether to suppress all output (default: FALSE). When TRUE, completely silences all messages, progress indicators, and Quarto rendering output. Useful for scripts and automated workflows. Overrides show_progress.

standalone Whether to embed all resources (CSS, JS, images, fonts) into a single self-contained HTML file (default: FALSE). When TRUE, sets Quarto's `embed-resources: true` so the output HTML can be shared without a web server. Implies `render = TRUE`.

Value

Invisibly returns the project object with build_info attached

Examples

```
## Not run:
# Generate and render dashboard
dashboard %>% generate_dashboard(render = TRUE, open = "browser")

# Generate without rendering (faster for quick iterations)
dashboard %>% generate_dashboard(render = FALSE)

# Incremental builds (skip unchanged pages)
dashboard %>% generate_dashboard(render = TRUE, incremental = TRUE)

# Preview specific page
dashboard %>% generate_dashboard(preview = "Analysis")

# Quiet mode for scripts
dashboard %>% generate_dashboard(render = FALSE, quiet = TRUE)

# Standalone HTML (single file, all resources embedded)
dashboard %>% generate_dashboard(standalone = TRUE)

## End(Not run)
```

---

generate_dashboards *Generate multiple dashboards*

Description

Generates a list of dashboard projects in batch, with progress tracking and error handling. Useful for generating many related dashboards (e.g., one per country, per topic, etc.) in a single workflow.

Usage

```
generate_dashboards(
  dashboards,
  render = TRUE,
  open = FALSE,
  continue_on_error = TRUE,
  show_progress = TRUE,
  quiet = FALSE,
  linked = FALSE
)
```

Arguments

dashboards	Named list of dashboard_project objects created with create_dashboard(). When linked = TRUE, the first dashboard is the main/parent and others are sub-dashboards that will be output into subdirectories of the main's docs folder.
render	Whether to render each dashboard to HTML (default TRUE)
open	Whether to open the main dashboard after generation (default FALSE)
continue_on_error	Continue generating remaining dashboards if one fails (default TRUE)
show_progress	Whether to show progress for each dashboard (default TRUE)
quiet	Whether to suppress output (default FALSE)
linked	Whether dashboards are linked (default FALSE). When TRUE: • First dashboard is treated as the main/parent dashboard • Other dashboards are output to subdirectories of main's docs folder • Use list names as subdirectory names (e.g., list(main = ..., US = ..., DE = ...)) • Click navigation like click_url_template = "\{iso2c\}/index.html" will work

Value

Invisibly returns a list of results, one per dashboard, containing:

- success: logical, whether generation succeeded
- title: dashboard title
- output_dir: output directory path
- error: error message if failed (only present on failure)
- duration: generation time in seconds

Examples

```
## Not run:
# Linked dashboards with map navigation
main_db <- create_dashboard("Main", output_dir = "project") %>%
```

```
  add_page("Map", data = summary_data,
           visualizations = create_viz() %>%
             add_viz(type = "map", click_url_template = "{iso2c}/index.html"))

us_db <- create_dashboard("US Details", output_dir = "project/US") %>%
  add_page("Analysis", data = us_data)

de_db <- create_dashboard("DE Details", output_dir = "project/DE") %>%
  add_page("Analysis", data = de_data)

# Generate with linked = TRUE - outputs go to project/docs/, project/docs/US/, etc.
generate_dashboards(
  list(main = main_db, US = us_db, DE = de_db),
  linked = TRUE
)

## End(Not run)
```

`html_accordion` *Create a collapsible accordion section*

Description

Create a collapsible accordion section

Usage

```
html_accordion(body, title = "Details")
```

Arguments

`body`	Accordion body content (character string).
`title`	Summary/header text. Default "Details".

Value

An htmltools tag object.

html_badge *Create a status badge*

Description

Create a status badge

Usage

```
html_badge(text, color = "primary")
```

Arguments

text	Badge text.
color	Badge color class: "success", "warning", "danger", "info", "primary" (default), or "secondary".

Value

An htmltools tag object.

html_card *Create a Bootstrap-style card*

Description

Create a Bootstrap-style card

Usage

```
html_card(body, title = NULL)
```

Arguments

body	Card body content (character string).
title	Optional card header title.

Value

An htmltools tag object.

html_divider *Create a horizontal divider*

Description

Returns a styled <hr> tag. For the default style, prefer markdown --- instead.

Usage

```
html_divider(style = "thick")
```

Arguments

style	Divider style: "thick", "dashed", or "dotted".

Value

An htmltools tag object.

html_iframe *Create an iframe embed*

Description

Create an iframe embed

Usage

```
html_iframe(url, height = "500px", width = "100%", style = NULL)
```

Arguments

url	URL to embed.
height	CSS height value. Default "500px".
width	CSS width value. Default "100%".
style	Optional additional CSS styles.

Value

An htmltools tag object.

`html_metric` *Create a metric card*

Description

Create a metric card

Usage

```
html_metric(
  value,
  title,
  icon = NULL,
  color = NULL,
  bg_color = NULL,
  text_color = NULL,
  gradient = TRUE,
  gradient_intensity = 0.45,
  value_prefix = NULL,
  value_suffix = NULL,
  border_radius = NULL,
  subtitle = NULL,
  aria_label = NULL
)
```

Arguments

`value`	The metric value to display.
`title`	Metric title.
`icon`	Optional icon name (e.g. "mdi:account"). Uses the iconify web component.
`color`	Optional accent color for left border.
`bg_color`	Optional background color (e.g. "#3498db").
`text_color`	Optional text color applied to the card (e.g. "#ffffff"). Also sets the icon color instead of the default `text-primary` class.
`value_prefix`	Optional string prepended to the displayed value.
`value_suffix`	Optional string appended to the displayed value.
`border_radius`	Optional CSS border-radius (e.g. "12px", "0").
`subtitle`	Optional subtitle text.
`aria_label`	Optional ARIA label for accessibility.

Value

An htmltools tag object.

html_spacer *Create a vertical spacer*

Description

Returns an htmltools div with the specified height. Use in dashboards to add vertical spacing between content blocks.

Usage

```
html_spacer(height = "1rem")
```

Arguments

height	CSS height value (e.g. "1rem", "20px"). Default "1rem".

Value

An htmltools tag object.

icon *Create iconify icon shortcode*

Description

Helper function to generate iconify icon shortcodes for use in pages and visualizations.

Usage

```
icon(icon_name)
```

Arguments

icon_name	Icon name in format "collection:name" (e.g., "ph:users-three")

Value

Iconify shortcode string

Examples

```
## Not run:
icon("ph:users-three")  # Returns iconify shortcode
icon("emojione:flag-for-united-states")  # Returns iconify shortcode

## End(Not run)
```

knit_print.content_collection

Knitr print method for content collections

Description

Automatically renders content collections as interactive visualizations when output in knitr documents (vignettes, pkgdown articles, R Markdown). If no data is attached to the collection, shows the structure instead.

Usage

```
## S3 method for class 'content_collection'
knit_print(x, ..., options = NULL)
```

Arguments

x	A content_collection or viz_collection object
...	Additional arguments (currently ignored)
options	Knitr chunk options (currently ignored)

Details

This method enables "show the viz" behavior in documents while preserving the structure print for console debugging. Simply output a collection with inline data to see the rendered visualization:

```
create_viz(data = mtcars)
  add_viz(type = "histogram", x_var = "mpg")
# Renders as interactive chart in documents!
```

Value

A knitr asis_output object containing the rendered HTML

knit_print.dashboard_project

Knitr print method for dashboard projects

Description

Automatically renders dashboard projects as a preview in knitr documents. Shows a combined view of all pages or the landing page.

Usage

```
## S3 method for class 'dashboard_project'
knit_print(x, ..., options = NULL)
```

Arguments

x	A dashboard_project
...	Additional arguments
options	Knitr chunk options

Value

A knitr asis_output object containing the rendered HTML

---

knit_print.page_object

Knitr print method for page objects

Description

Automatically renders page objects as interactive content in knitr documents. Converts the page to a content collection and renders it.

Usage

```
## S3 method for class 'page_object'
knit_print(x, ..., options = NULL)
```

Arguments

x	A page_object
...	Additional arguments
options	Knitr chunk options

Value

A knitr asis_output object containing the rendered HTML

md_text *Create multi-line markdown text content*

Description

Helper function to create readable multi-line markdown text content for pages. Automatically handles line breaks and formatting for better readability.

Usage

```
md_text(..., sep = "\n")
```

Arguments

...	Text content as separate arguments or character vectors
sep	Separator to use when joining text (default: "\n" for newlines). Use "" for no separator.

Value

Single character string with proper line breaks

Examples

```
## Not run:
# Method 1: Separate arguments (default: newlines between)
text_content <- md_text(
  "# Welcome",
  "",
  "This is a multi-line text block.",
  "",
  "## Features",
  "- Feature 1",
  "- Feature 2"
)

# Method 2: Character vectors
lines <- c("# About", "", "This is about our study.")
text_content <- md_text(lines)

# Method 3: Combine without newlines
combined <- md_text(text1, text2, text3, sep = "")

# Use in add_page
add_page("About", text = text_content)

## End(Not run)
```

merge_collections *Merge two content/viz collections*

Description

Internal function to merge two collections into one.

Usage

```
merge_collections(c1, c2)
```

Arguments

c1	First collection
c2	Second collection

Value

Merged content_collection

modal_content *Create Modal Content Container*

Description

Creates a hidden div that contains the content to be displayed in a modal. The content will be shown when a link with matching modal_id is clicked.

Usage

```
modal_content(modal_id, ..., title = NULL, image = NULL, text = NULL)
```

Arguments

modal_id	Unique ID for this modal content
...	Content to display in modal (images, text, HTML)
title	Optional title to display at top of modal
image	Optional image path or URL to display
text	Optional text/HTML content to display below image

Value

HTML div element

Examples

```
## Not run:
# Simple text modal
modal_content(
  modal_id = "info",
  title = "Information",
  text = "This is some important information."
)

# Modal with image and text
modal_content(
  modal_id = "chart1",
  title = "Sales Chart",
  image = "charts/sales.png",
  text = "This chart shows sales trends over the past year."
)

# Custom content
modal_content(
  modal_id = "custom",
  htmltools::tags$h2("Custom Title"),
  htmltools::tags$img(src = "image.jpg"),
  htmltools::tags$p("Description text"),
  htmltools::tags$ul(
    htmltools::tags$li("Point 1"),
    htmltools::tags$li("Point 2")
  )
)

## End(Not run)
```

---

modal_link *Create Modal Link*

Description

Creates a hyperlink that opens content in a modal dialog instead of navigating to a new page. You can also use regular markdown syntax: `[Link Text](#modal-id)` and it will automatically open as a modal.

Usage

```
modal_link(text, modal_id, class = NULL)
```

Arguments

text	Link text to display
modal_id	ID of the modal content div
class	Additional CSS classes for the link

Value

HTML link element

Examples

```
## Not run:
modal_link("View Details", "details-modal")
modal_link("See Chart", "chart1", class = "btn btn-primary")

# Or in markdown:
# [View Details](#details-modal)

## End(Not run)
```

`navbar_menu` *Create a navbar dropdown menu*

Description

Creates a dropdown menu in the navbar without requiring sidebar groups. This is a simple nested menu structure.

Usage

```
navbar_menu(text, pages, icon = NULL, align = c("left", "right"))
```

Arguments

`text`	Display text for the dropdown menu button
`pages`	Character vector of page names to include in the dropdown
`icon`	Optional icon for the menu button
`align`	Where to place the menu in the navbar: "left" (default) or "right"

Value

List containing navbar menu configuration

Examples

```
## Not run:
# Create a simple dropdown menu (left-aligned by default)
dimensions_menu <- navbar_menu(
  text = "Dimensions",
  pages = c("Strategic Information", "Critical Information"),
  icon = "ph:book"
)

# Create a right-aligned menu
```

```
more_info_menu <- navbar_menu(
  text = "More Info",
  pages = c("About", "Wave 1"),
  icon = "ph:info",
  align = "right"
)

dashboard <- create_dashboard(
  navbar_sections = list(dimensions_menu, more_info_menu)
)

## End(Not run)
```

`navbar_section` *Create a navbar section for hybrid navigation*

Description

Helper function to create a navbar section that links to a sidebar group for hybrid navigation. This creates dropdown-style navigation.

Usage

```
navbar_section(text, sidebar_id, icon = NULL)
```

Arguments

`text`	Display text for the navbar item
`sidebar_id`	ID of the sidebar group to link to
`icon`	Optional icon for the navbar item

Value

List containing navbar section configuration

Examples

```
## Not run:
# Create navbar sections that link to sidebar groups
analysis_section <- navbar_section("Analysis", "analysis", "ph:chart-bar")
reference_section <- navbar_section("Reference", "reference", "ph:book")

## End(Not run)
```

preview *Preview any dashboardr object*

Description

Universal preview function that renders any dashboardr object to HTML and displays it in the RStudio Viewer pane or browser. Supports dashboard_project, page_object, content_collection, viz_collection, and individual content_block objects. Useful for developing and testing dashboards without building the entire project.

Usage

```
preview(
  collection,
  title = "Preview",
  open = TRUE,
  clean = FALSE,
  quarto = FALSE,
  theme = "cosmo",
  path = NULL,
  page = NULL,
  debug = FALSE,
  output = c("viewer", "widget")
)
```

Arguments

collection	A dashboardr object to preview. Can be any of: • dashboard_project - previews all pages with full styling • page_object - previews a single page • content_collection or viz_collection - previews content/visualizations • content_block - previews a single content block (text, callout, etc.) For collections with visualizations, data must be attached via the data parameter in create_viz()/create_content().
title	Optional title for the preview document (default: "Preview")
open	Whether to automatically open the result in viewer/browser (default: TRUE)
clean	Whether to clean up temporary files after viewing (default: FALSE)
quarto	Whether to use Quarto for rendering (default: FALSE). When FALSE (default), uses direct R rendering which is faster and doesn't require Quarto. When TRUE, creates a full Quarto document (useful for testing tabsets/icons).
theme	Bootstrap theme for Quarto preview (default: "cosmo", only used when quarto=TRUE)
path	Optional path to save the preview. If NULL (default), uses a temp directory. Can be a directory path (preview.html will be created inside) or a file path ending in .html.

page	Optional page name to preview (only used for dashboard_project objects). When NULL, previews all pages. When specified, previews only the named page.
debug	Whether to show debug messages like file paths (default: FALSE).
output	Output mode: "viewer" (default) opens in RStudio viewer/browser, "widget" returns an htmltools widget that can be saved as self-contained HTML with `save_widget()` or embedded in R Markdown/Quarto documents.

Details

The preview function has two modes:

Direct mode (quarto = FALSE, default):

- Directly calls visualization functions with the attached data
- Renders all content blocks (text, callouts, cards, tables, etc.)
- Includes interactive elements (inputs, modals) with CDN dependencies
- Wraps results in a styled HTML page using htmltools
- Fast and doesn't require Quarto installation
- Best for quick iteration during development

Quarto mode (quarto = TRUE):

- Creates a temporary Quarto document
- Renders with full Quarto features (tabsets, icons, theming)
- Applies dashboard styling (navbar colors, fonts, tabset themes)
- Requires Quarto to be installed
- Best for testing final dashboard appearance

Supported content types: Text/display: text, html, quote, badge, metric Layout: divider, spacer, card, accordion Media: image, video, iframe Tables: gt, reactable, DT, table Interactive: input, input_row, modal Value boxes: value_box, value_box_row

Value

For output="viewer": invisibly returns the path to the generated HTML file. For output="widget": returns a dashboardr_widget object that can be saved or embedded.

Examples

```
## Not run:
# Preview a dashboard project
my_dashboard %>% preview()
my_dashboard %>% preview(page = "Analysis")  # Preview specific page
my_dashboard %>% preview(quarto = TRUE)  # Full Quarto rendering

# Preview a page object
create_page("Analysis", data = mtcars) %>%
  add_viz(type = "histogram", x_var = "mpg") %>%
  preview()
```

```
# Preview a visualization collection
create_viz(data = mtcars) %>%
  add_viz(type = "histogram", x_var = "mpg", title = "MPG Distribution") %>%
  preview()

# Preview with Quarto for full features (required for tabsets!)
create_viz(data = mtcars) %>%
  add_viz(type = "histogram", x_var = "mpg", tabgroup = "MPG") %>%
  add_viz(type = "histogram", x_var = "hp", tabgroup = "HP") %>%
  preview(quarto = TRUE)

# Preview content blocks
create_content() %>%
  add_text("# Hello World") %>%
  add_callout("Important note", type = "tip") %>%
  preview()

# Save preview to specific location
my_viz %>% preview(path = "~/Desktop/my_preview.html")

# Preview without opening (just render)
html_path <- my_viz %>% preview(open = FALSE)

# Return as widget (for embedding in R Markdown or saving as self-contained HTML)
widget <- my_viz %>% preview(output = "widget")
htmltools::save_html(widget, "my_chart.html", selfcontained = TRUE)

# In R Markdown/Quarto, widgets display inline automatically
my_viz %>% preview(output = "widget")

## End(Not run)
```

`print.dashboard_project` *Print Dashboard Project*

Description

Displays a comprehensive summary of a dashboard project, including metadata, features, pages, visualizations, and integrations.

Usage

```
## S3 method for class 'dashboard_project'
print(x, ...)
```

Arguments

x	A dashboard_project object created by `create_dashboard`.
...	Additional arguments (currently ignored).

Details

The print method displays:

- Project metadata (title, author, description)
- Output directory
- Enabled features (sidebar, search, themes, Shiny, Observable)
- Integrations (GitHub, Twitter, LinkedIn, Analytics)
- Page structure with properties:
  - Landing page indicator
  - Loading overlay indicator
  - Right-aligned navbar indicator
  - Associated datasets
  - Nested visualization hierarchies

Value

Invisibly returns the input object `x`.

`print.dashboardr_tooltip` *Print method for tooltip configurations*

Description

Print method for tooltip configurations

Usage

```
## S3 method for class 'dashboardr_tooltip'
print(x, ...)
```

Arguments

`x`	A dashboardr_tooltip object
`...`	Ignored

print.dashboardr_widget *Print method for dashboardr_widget - opens in viewer*

Description

Print method for dashboardr_widget - opens in viewer

Usage

```
## S3 method for class 'dashboardr_widget'
print(x, ...)
```

Arguments

x	A dashboardr_widget object to print
...	Additional arguments (currently ignored)

print.page_object *Print method for page objects*

Description

Print method for page objects

Usage

```
## S3 method for class 'page_object'
print(x, ...)
```

Arguments

x	A page_object to print
...	Additional arguments (currently ignored)

print.viz_collection *Print Visualization Collection*

Description

Displays a formatted summary of a visualization collection, including hierarchical tabgroup structure, visualization types, titles, filters, and defaults.

Usage

```
## S3 method for class 'viz_collection'
print(x, render = FALSE, check = FALSE, ...)
```

Arguments

x	A viz_collection object created by create_viz.
render	If TRUE and data is attached, opens a preview in the viewer instead of showing the structure. Default is FALSE.
check	Logical. If TRUE, validates all visualization specs before printing. Useful for catching errors early before attempting to render.
...	Additional arguments (currently ignored).

Details

The print method displays:

- Total number of visualizations
- Default parameters (if set)
- Hierarchical tree structure showing tabgroup organization
- Visualization types with emoji indicators
- Filter status for each visualization

Use `print(x, render = TRUE)` to open a preview in the viewer instead of showing the structure. This is useful for quick visualization in the console.

Use `print(x, check = TRUE)` to validate all visualization specs before printing. This catches missing required parameters and invalid column names early, providing clearer error messages than Quarto rendering errors.

Value

Invisibly returns the input object x.

publish_dashboard *Publish dashboard to GitHub Pages*

Description

This function automates the process of publishing a dashboard to GitHub Pages. It handles git initialization, .gitignore setup, GitHub repository creation, and GitHub Pages configuration using usethis functions.

Usage

```
publish_dashboard(
  message = "Initial commit",
  restart = FALSE,
  organisation = NULL,
  private = NULL,
  protocol = c("https", "ssh"),
  branch = usethis::git_default_branch(),
  path = "/docs",
  ask = TRUE,
  ...
)
```

Arguments

message	Initial commit message (default: "Initial commit")
restart	Whether to restart RStudio after git initialization (default: FALSE)
organisation	GitHub organisation name (optional, for org repositories)
private	Whether to create a private repository. When NULL (default) and ask = TRUE, you will be prompted interactively. Set to TRUE or FALSE to skip the prompt.
protocol	Transfer protocol: "https" or "ssh" (default: "https")
branch	Branch to deploy from (default: uses git default branch)
path	Path containing the site files (default: "/docs")
ask	Whether to use the interactive confirmation workflow (default: TRUE). When TRUE, guides you through file review and confirmation steps. Set to FALSE to skip all prompts (not recommended for first-time use).
...	Additional arguments passed to usethis::use_github()

Details

When ask = TRUE (the default), the function guides you through a 3-step interactive confirmation process:

1. **File Review**: Shows you the files that will be published and opens a folder so you can verify nothing unintended is included
2. **Repository Privacy**: Asks whether to create a private or public repository
3. **Confirm Publish**: Final confirmation before publishing to GitHub

Value

Invisibly returns TRUE if published successfully, FALSE if cancelled

What Gets Published

Typically, you only need to publish:

- The `docs/` folder (auto-generated HTML, CSS, JS files)
- Optionally, your R scripts (just for reproducibility)

By default, common data file extensions (.csv, .rds, .xlsx, .sav, .dta) are automatically excluded via .gitignore. Use `usethis::use_git_ignore()` to exclude additional files you don't want to publish.

Examples

```
## Not run:
# After generating a dashboard, navigate to the dashboard directory
# and publish it (interactive mode):
setwd("my_dashboard")
publish_dashboard()

# Publish to an organization
publish_dashboard(organisation = "my-org")

# Create a private repository (skip privacy prompt)
publish_dashboard(private = TRUE)

# Skip all prompts (use with caution)
publish_dashboard(ask = FALSE, private = FALSE)

## End(Not run)
```

---

render_input *Render an input widget*

Description

Creates HTML for various input widgets that filter Highcharts visualizations.

Usage

```
render_input(
  input_id,
  label = NULL,
  type = c("select_multiple", "select_single", "checkbox", "radio", "switch", "slider",
    "text", "number", "button_group", "date", "daterange"),
  filter_var,
  options = NULL,
  options_from = NULL,
```

```
  default_selected = NULL,
  placeholder = "Select...",
  width = "300px",
  align = c("center", "left", "right"),
  min = 0,
  max = 100,
  step = 1,
  value = NULL,
  show_value = TRUE,
  inline = TRUE,
  stacked = FALSE,
  stacked_align = c("center", "left", "right"),
  group_align = c("left", "center", "right"),
  ncol = NULL,
  nrow = NULL,
  columns = NULL,
  toggle_series = NULL,
  override = FALSE,
  labels = NULL,
  size = c("md", "sm", "lg"),
  help = NULL,
  disabled = FALSE,
  icons = NULL,
  linked_child_id = NULL,
  options_by_parent = NULL
)
```

Arguments

input_id	Unique ID for this input widget
label	Optional label displayed above the input
type	Input type: "select_multiple", "select_single", "checkbox", "radio", "switch", "slider", "text", "number", "button_group", "date", or "daterange"
filter_var	The variable name to filter by (matches Highcharts series names)
options	Character vector of options to display (for select/checkbox/radio/button_group). Can also be a named list for grouped options in selects.
options_from	Column name in page data to auto-populate options from
default_selected	Character vector of initially selected values
placeholder	Placeholder text when nothing is selected (for selects/text)
width	CSS width for the input
align	Alignment: "center", "left", or "right"
min	Minimum value (for slider/number)
max	Maximum value (for slider/number)
step	Step increment (for slider/number)

value	Initial value (for slider/switch/text/number)
show_value	Whether to show the current value (for slider)
inline	Whether to display options inline (for checkbox/radio)
stacked	Whether to stack options vertically (for checkbox/radio). Default FALSE.
stacked_align	Alignment when stacked: "center" (default), "left", or "right"
group_align	Alignment for option groups: "left" (default), "center", or "right"
ncol	Number of columns for grid layout of options
nrow	Number of rows for grid layout of options
columns	Column configuration for grid layout
toggle_series	For switch type: name of the series to toggle on/off
override	For switch type: if TRUE, switch overrides other filters for this series
labels	Custom labels for slider ticks (character vector)
size	Size variant: "sm", "md" (default), or "lg"
help	Help text displayed below the input
disabled	Whether the input is disabled
icons	Optional character vector of Iconify icon names (e.g., "ph:calendar", "ph:users-three"). Must be same length as options. When provided, icons are rendered before option text for radio and button_group types.
linked_child_id	ID of linked child input for cascading inputs
options_by_parent	Named list mapping parent values to child options

Value

HTML output (invisible)

`render_input_row` *Render a row of input widgets*

Description

Creates HTML for a horizontal row of input widgets.

Usage

```
render_input_row(inputs, style = "boxed", align = "center")
```

Arguments

inputs	List of input specifications (each should have the same parameters as render_input)
style	Visual style: "boxed" (default) or "inline" (compact)
align	Alignment: "center" (default), "left", or "right"

Value

HTML output

`render_value_box` *Render a single value box*

Description

Render a single value box

Usage

```
render_value_box(
  title,
  value,
  bg_color = "#2c3e50",
  logo_url = NULL,
  logo_text = NULL,
  aria_label = NULL
)
```

Arguments

`title`	Box title (small text above value)
`value`	Main value to display (large text)
`bg_color`	Background color (hex code), default "#2c3e50"
`logo_url`	Optional URL or path to logo image
`logo_text`	Optional text to display as logo (if no logo_url)
`aria_label`	Optional ARIA label for accessibility.

Value

An htmltools tag object.

render_value_box_row *Render a row of value boxes*

Description

Render a row of value boxes

Usage

```
render_value_box_row(boxes)
```

Arguments

boxes	List of value box specifications, each containing title, value, bg_color, logo_url, logo_text

Value

An htmltools tag object.

render_viz_html *Render a viz result as raw HTML*

Description

In `results='asis'` chunks (e.g. when using show_when_open), bare htmlwidget objects are NOT rendered by knitr. This helper converts the widget (or tagList from `.embed_cross_tab`) to HTML and `cat()`s it so it appears in the output.

Usage

```
render_viz_html(result)
```

Arguments

result	A highcharter object, htmlwidget, shiny.tag, or shiny.tag.list

Value

Called for its side-effect (`cat()`).

Examples

```
## Not run:
# In a QMD chunk with results='asis':
show_when_open('{"var":"year","op":"eq","val":"2024"}')
result <- viz_bar(data = df, x_var = "category")
render_viz_html(result)
show_when_close()

## End(Not run)
```

`save_widget` *Save widget as self-contained HTML*

Description

Save widget as self-contained HTML

Usage

```
save_widget(widget, file, selfcontained = TRUE)
```

Arguments

`widget`	A dashboardr widget created with preview(output = "widget")
`file`	Path to save the HTML file
`selfcontained`	Whether to embed all dependencies (default: TRUE)

Value

Invisibly returns the file path

Examples

```
## Not run:
widget <- my_viz %>% preview(output = "widget")
save_widget(widget, "my_chart.html")

## End(Not run)
```

`set_tabgroup_labels` *Set or update tabgroup display labels*

Description

Updates the display labels for tab groups in a visualization collection. Useful when you want to change the section headers after creating the collection.

Usage

```
set_tabgroup_labels(viz_collection, labels = NULL, ...)
```

Arguments

`viz_collection`	A viz_collection object
`labels`	Named character vector or list mapping tabgroup IDs to labels (deprecated, use ... instead)
`...`	Named arguments where names are tabgroup IDs and values are display labels

Value

The updated viz_collection

Examples

```
## Not run:
# New style: direct key-value pairs (recommended)
vizzes <- create_viz() %>%
  add_viz(type = "heatmap", tabgroup = "demo") %>%
  set_tabgroup_labels(demo = "Demographic Breakdowns", age = "Age Groups")

# Old style: still supported for backwards compatibility
vizzes <- create_viz() %>%
  add_viz(type = "heatmap", tabgroup = "demo") %>%
  set_tabgroup_labels(list(demo = "Demographic Breakdowns"))

## End(Not run)
```

set_tabgroup_labels.page_object *Set tabgroup labels for a page*

Description

Customize tab labels with icons or different text.

Usage

```
set_tabgroup_labels.page_object(page, ...)
```

Arguments

page	A page_object
...	Named arguments where names are tabgroup IDs and values are display labels

Value

The updated page_object

Examples

```
## Not run:
create_page("Analysis", data = gss, type = "bar") %>%
  add_viz(x_var = "degree", tabgroup = "demographics") %>%
  add_viz(x_var = "happy", tabgroup = "wellbeing") %>%
  set_tabgroup_labels(
    demographics = "{{< iconify ph:users-fill >}} Demographics",
    wellbeing = "{{< iconify ph:heart-fill >}} Wellbeing"
  )

## End(Not run)
```

show_structure *Show collection structure (even with data attached)*

Description

Forces display of the collection structure instead of rendering visualizations. Useful when you want to inspect the structure of a collection that has data attached, or when documenting the collection's organization.

Usage

```
show_structure(x)
```

Arguments

x	A content_collection or viz_collection object

Value

In knitr: formatted HTML output. In console: invisible(x) after printing.

Examples

```
## Not run:
# In a vignette or R Markdown, pipe into print() to see the tree
# even when data is attached:
create_viz(data = mtcars, type = "bar") %>%
  add_viz(x_var = "cyl", title = "Cylinders") %>%
  print()

## End(Not run)
```

---

show_when_close *Close a conditional-visibility wrapper*

Description

Emits the closing that matches show_when_open.

Usage

```
show_when_close()
```

Value

Called for its side-effect (cat()).

---

show_when_open *Open a conditional-visibility wrapper*

Description

Emits an opening with the data-show-when attribute so that show_when.js can show/hide the enclosed content based on input state.

Usage

```
show_when_open(condition_json)
```

Arguments

condition_json	A JSON string describing the condition (e.g. '{"var":"time_period","op":"in","val":["Wave 1","Wave 2"]}').

Details

This is used in generated .qmd chunks – users typically do not need to call it directly.

Value

Called for its side-effect (cat()).

---

showcase_dashboard *Generate a showcase dashboard demonstrating all dashboardr features.*

Description

This function creates and renders a comprehensive showcase dashboard that demonstrates the full breadth of the dashboardr package. It includes multiple visualization types, tabset grouping, standalone charts, and various page layouts.

Usage

```
showcase_dashboard(directory = "showcase_dashboard", open = "browser")
```

Arguments

directory	Character string. Directory where the dashboard files will be created. Defaults to "showcase_dashboard". Quarto will render HTML to directory/docs/.
open	Logical or character. Whether to open the dashboard after rendering. Use TRUE or "browser" to open in browser (default), FALSE to not open. Default is "browser".

Details

The showcase dashboard uses General Social Survey (GSS) data to demonstrate:

- Multiple tabset groups (Demographics, Politics, Social Issues)
- Stacked bar charts with custom styling
- Heatmaps with custom color palettes
- Standalone charts without tabsets
- Text-only pages with card layouts
- Mixed content pages (text + visualizations)
- Custom icons throughout
- All advanced dashboard features

This dashboard is more comprehensive than the tutorial dashboard and showcases the full power of dashboardr for creating complex, multi-page dashboards.

Value

Invisibly returns the dashboard_project object.

Examples

```
## Not run:
# Run the showcase dashboard (requires Quarto CLI and 'gssr' package)
showcase_dashboard()

# Specify custom directory
showcase_dashboard(directory = "my_showcase")

# Don't open browser
showcase_dashboard(open = FALSE)

## End(Not run)
```

---

sidebar_group *Create a sidebar group for hybrid navigation*

Description

Helper function to create a sidebar group configuration for use with hybrid navigation. Each group can have its own styling and contains a list of pages.

Usage

```
sidebar_group(
  id,
  title,
  pages,
  style = NULL,
  background = NULL,
  foreground = NULL,
  border = NULL,
  alignment = NULL,
  collapse_level = NULL,
  pinned = NULL,
  tools = NULL
)
```

Arguments

id	Unique identifier for the sidebar group
title	Display title for the sidebar group
pages	Character vector of page names to include in this group
style	Sidebar style (docked, floating, etc.) (optional)

background	Background color (optional)
foreground	Foreground color (optional)
border	Show border (optional)
alignment	Alignment (left, right) (optional)
collapse_level	Collapse level for navigation (optional)
pinned	Whether sidebar is pinned (optional)
tools	List of tools to add to sidebar (optional)

Value

List containing sidebar group configuration

Examples

```
## Not run:
# Create a sidebar group for analysis pages
analysis_group <- sidebar_group(
  id = "analysis",
  title = "Data Analysis",
  pages = c("overview", "demographics", "findings"),
  style = "docked",
  background = "light"
)

## End(Not run)
```

---

spec_viz *Create a single visualization specification*

Description

Helper function to create individual viz specs that can be combined into a list or used directly in add_page().

Usage

```
spec_viz(type, ..., tabgroup = NULL, title = NULL)
```

Arguments

type	Visualization type
...	Additional parameters
tabgroup	Optional group ID
title	Display title

Value

A list containing the visualization specification

Examples

```
## Not run:
viz1 <- spec_viz(type = "heatmap", x_var = "party", y_var = "ideology")
viz2 <- spec_viz(type = "histogram", x_var = "age")
page_viz <- list(viz1, viz2)

## End(Not run)
```

---

text_lines *Create text content from a character vector*

Description

Alternative helper for creating text content from existing character vectors.

Usage

```
text_lines(lines)
```

Arguments

lines	Character vector of text lines

Value

Single character string with proper line breaks

Examples

```
## Not run:
lines <- c("# Title", "", "Content here")
text_content <- text_lines(lines)
add_page("Page", text = text_content)

## End(Not run)
```

theme_academic *Apply a Professional Academic Theme to Dashboard*

Description

Returns a clean, professional theme suitable for academic and research dashboards. Uses neutral colors and excellent typography for maximum readability. All parameters can be overridden to customize the theme.

Usage

```
theme_academic(accent_color = "#2563eb", navbar_style = "dark", ...)
```

Arguments

accent_color	Primary accent color (hex code). Default: "#2563eb" (blue)
navbar_style	Style of the navbar. Options: "dark" (default), "light"
...	Additional theme parameters to override defaults. Can include any styling parameter like navbar_bg_color, navbar_text_color, navbar_text_hover_color, mainfont, fontsize, etc.

Details

The academic theme provides:

- Clean, neutral color scheme
- Professional typography (Fira Sans + Source Code Pro)
- High readability settings
- Suitable for any academic institution

Value

A named list of theme parameters that can be unpacked into create_dashboard()

Examples

```
## Not run:
# Use default academic theme
dashboard <- create_dashboard("academic_dashboard", "Research Dashboard") %>%
  apply_theme(theme_academic()) %>%
  add_page("Home", text = "# Welcome")

# Custom accent color and font
dashboard <- create_dashboard("my_university", "University Research") %>%
  apply_theme(theme_academic(
    accent_color = "#8B0000",
    mainfont = "Roboto",
```

```
    fontsize = "17px"
  ))

## End(Not run)
```

`theme_ascor` *Apply ASCoR/UvA Theme to Dashboard*

Description

Returns a list of styling parameters that apply University of Amsterdam and ASCoR (Amsterdam School of Communication Research) branding to a dashboard. Can be used with `apply_theme()` for piping or unpacked into `create_dashboard()`. All parameters can be overridden to customize the theme.

Usage

```
theme_ascor(navbar_style = "dark", ...)
```

Arguments

`navbar_style`	Style of the navbar. Options: "dark" (default), "light". Dark style works best with UvA red.
`...`	Additional theme parameters to override defaults. Can include any styling parameter like `navbar_bg_color`, `navbar_text_color`, `navbar_text_hover_color`, `mainfont`, `fontsize`, etc.

Details

The ASCoR theme includes:

- UvA red (#CB0D0D) as primary brand color
- Professional Fira Sans font for body text
- Fira Code for code blocks
- Optimal readability settings
- Clean, academic styling

Value

A named list of theme parameters that can be unpacked into `create_dashboard()`

Examples

```
## Not run:
# Method 1: Use default ASCoR theme
dashboard <- create_dashboard("my_dashboard", "My Research Dashboard") %>%
  apply_theme(theme_ascor()) %>%
  add_page("Home", text = "# Welcome", is_landing_page = TRUE)

# Method 2: Override specific parameters
dashboard <- create_dashboard("custom", "Custom ASCoR Dashboard") %>%
  apply_theme(theme_ascor(
    fontsize = "18px",
    max_width = "1400px",
    mainfont = "Inter"
  ))

## End(Not run)
```

---

theme_clean *Apply a Clean Theme to Dashboard*

Description

Returns an ultra-clean, minimalist theme with maximum focus on content. Perfect for portfolios, reports, and clean presentations. All parameters can be overridden to customize the theme.

Usage

```
theme_clean(...)
```

Arguments

...	Additional theme parameters to override defaults. Can include any styling parameter like navbar_bg_color, navbar_text_color, navbar_text_hover_color, mainfont, fontsize, etc.

Value

A named list of theme parameters

Examples

```
## Not run:
# Use default clean theme
dashboard <- create_dashboard("clean_dashboard", "Clean Report") %>%
  apply_theme(theme_clean()) %>%
  add_page("Report", text = "# Executive Summary")

# Customize with wider layout and different font
dashboard <- create_dashboard("custom_clean", "Custom Report") %>%
```

```
  apply_theme(theme_clean(
    mainfont = "Inter",
    max_width = "1200px",
    fontsize = "18px"
  ))

## End(Not run)
```

---

theme_modern *Apply a Modern Tech Theme to Dashboard*

Description

Returns a sleek, modern theme suitable for tech companies and data science teams. Features bold colors and contemporary typography. All parameters can be overridden.

Usage

```
theme_modern(style = c("blue", "purple", "green", "orange", "white"), ...)
```

Arguments

style	Style variant. Options: "blue" (default), "purple", "green", "orange", "white"
...	Additional theme parameters to override defaults. Can include any styling parameter like navbar_bg_color, navbar_text_color, navbar_text_hover_color, mainfont, fontsize, etc.

Value

A named list of theme parameters

Examples

```
## Not run:
# Use default modern blue theme
dashboard <- create_dashboard("tech_dashboard", "Data Science Dashboard") %>%
  apply_theme(theme_modern()) %>%
  add_page("Analytics", visualizations = my_viz)

# Purple variant with custom font
dashboard <- create_dashboard("purple_dashboard", "Analytics Dashboard") %>%
  apply_theme(theme_modern(style = "purple", mainfont = "Inter", fontsize = "18px")) %>%
  add_page("Data", data = my_data)

# White navbar
dashboard <- create_dashboard("clean_dashboard", "Clean Dashboard") %>%
  apply_theme(theme_modern(style = "white")) %>%
  add_page("Home", text = "# Welcome")

## End(Not run)
```

theme_uva *Apply UvA Theme to Dashboard (Alias)*

Description

Alias for `theme_ascor()`. Returns University of Amsterdam branding parameters.

Usage

```
theme_uva(navbar_style = "dark")
```

Arguments

`navbar_style`	Style of the navbar. Options: "dark" (default), "light". Dark style works best with UvA red.

Value

A named list of theme parameters

Examples

```
## Not run:
# Pipe UvA theme into dashboard
dashboard <- create_dashboard("uva_dashboard", "UvA Research Dashboard") %>%
  apply_theme(theme_uva()) %>%
  add_page("Home", text = "# Welcome")

## End(Not run)
```

tooltip *Create a Tooltip Configuration*

Description

Creates a tooltip configuration object for use with dashboardr visualization functions. This provides a unified way to customize tooltips across all chart types with full access to Highcharts tooltip options.

Usage

```
tooltip(
  format = NULL,
  prefix = NULL,
  suffix = NULL,
  header = NULL,
  shared = FALSE,
  style = NULL,
  backgroundColor = NULL,
  borderColor = NULL,
  borderRadius = NULL,
  borderWidth = NULL,
  shadow = TRUE,
  enabled = TRUE,
  followPointer = NULL,
  outside = NULL,
  ...
)
```

Arguments

format	Character. Format string with {placeholders}. Available placeholders vary by chart type: • {value} - Primary value (all charts) • {category} - X-axis category (bar, histogram, stackedbar) • {x} - X value (scatter, heatmap) • {y} - Y value (scatter, heatmap) • {name} - Point/series name (all charts) • {series} - Series name (grouped charts) • {percent} - Percentage (percent-type charts)
prefix	Character. Text prepended to the value. Shortcut for simple customization.
suffix	Character. Text appended to the value. Shortcut for simple customization.
header	Character or FALSE. Header format string, or FALSE to hide the header.
shared	Logical. If TRUE, shows a shared tooltip for all series at the same x-value. Default is FALSE.
style	Named list. CSS styles for the tooltip text, e.g., list(fontSize = "14px", fontWeight = "bold").
backgroundColor	Character. Background color for the tooltip (e.g., "#f5f5f5").
borderColor	Character. Border color for the tooltip.
borderRadius	Numeric. Corner radius in pixels.
borderWidth	Numeric. Border width in pixels.
shadow	Logical. Whether to show a shadow behind the tooltip. Default is TRUE.
enabled	Logical. Whether tooltips are enabled. Default is TRUE.

followPointer	Logical. Whether the tooltip should follow the mouse pointer.
outside	Logical. Whether to render the tooltip outside the chart SVG.
...	Additional Highcharts tooltip options passed directly to hc_tooltip().

Value

A dashboardr_tooltip object that can be passed to any viz_* function's tooltip parameter.

See Also

viz_bar, viz_scatter, viz_histogram

Examples

```
# Simple suffix
tooltip(suffix = "%")

# Custom format string
tooltip(format = "{category}: {value} respondents")

# Full styling
tooltip(
  format = "<b>{category}</b><br/>Count: {value}",
  backgroundColor = "#f5f5f5",
  borderColor = "#999",
  borderRadius = 8,
  style = list(fontSize = "14px")
)

# Shared tooltip for grouped charts
tooltip(shared = TRUE, format = "{series}: {value}")
```

---

tutorial_dashboard *Generate a tutorial dashboard.*

Description

This function creates and renders a detailed tutorial dashboard showcasing various features of the dashboardr package. It includes examples of stacked bar charts, heatmaps, multiple pages, and custom components.

Usage

```
tutorial_dashboard(directory = "tutorial_dashboard", open = "browser")
```

Arguments

directory	Character string. Directory where the dashboard files will be created. Defaults to "tutorial_dashboard". Quarto will render HTML to directory/docs/.
open	Logical or character. Whether to open the dashboard after rendering. Use TRUE or "browser" to open in browser (default), FALSE to not open. Default is "browser".

Details

The dashboard uses data from the General Social Survey (GSS) to demonstrate visualization and layout options.

Value

Invisibly returns the dashboard_project object.

Examples

```
## Not run:
# Run the tutorial dashboard (requires Quarto CLI and 'gssr' package)
tutorial_dashboard()

# Specify custom directory
tutorial_dashboard(directory = "my_tutorial")

# Don't open browser
tutorial_dashboard(open = FALSE)

## End(Not run)
```

---

update_dashboard *Update dashboard on GitHub*

Description

Convenience function to add, commit, and push changes to GitHub. Works from the current working directory.

Usage

```
update_dashboard(files = ".", message = "Update dashboard", ask = TRUE)
```

Arguments

files	Files to add. Can be: • "." to add all changes (default) • A character vector of specific file paths • A glob pattern (e.g., ".R", "docs/")
message	Commit message (default: "Update dashboard")
ask	Whether to ask for confirmation before pushing (default: TRUE)

Value

Invisibly returns TRUE

Examples

```
## Not run:
# Update all changes (will ask for confirmation)
update_dashboard()

# Update with custom message
update_dashboard(message = "Fix navbar styling")

# Update specific files
update_dashboard(files = c("docs/index.html", "docs/styles.css"))

# Skip confirmation prompt
update_dashboard(ask = FALSE)

## End(Not run)
```

---

validate_specs *Validate visualization specifications in a collection*

Description

Checks all visualization specs in a collection for common errors before rendering. This includes verifying required parameters are present and that specified column names exist in the data.

Usage

```
validate_specs(collection, verbose = TRUE, data = NULL)
```

Arguments

collection	A content_collection, viz_collection, page_object, or dashboard_project
verbose	Logical. If TRUE (default), prints validation results to console. If FALSE, returns silently with results as attributes.
data	Optional data frame to validate column names against. If NULL, uses data attached to the collection.

Details

This function is called automatically by `preview()` before rendering. You can also call it manually to check your visualizations before attempting to render, which provides clearer error messages than Quarto rendering errors.

Validation checks include:

- Required parameters for each visualization type (e.g., x_var for bar charts)
- Column existence in the data (when data is available)
- Suggestions for typos in column names

Value

Invisibly returns TRUE if all specs are valid, FALSE otherwise. When FALSE, the return value has an "issues" attribute containing details about validation errors.

Examples

```
## Not run:
# Create a collection with an error (missing required params)
# stackedbar requires either (x_var + stack_var) OR x_vars
viz <- create_viz(data = mtcars) %>%
  add_viz(type = "stackedbar", x_var = "cyl")  # Missing stack_var or x_vars

# Validate before previewing - will show helpful error
validate_specs(viz)

# Use in print with check parameter
print(viz, check = TRUE)

# Programmatic validation (silent)
result <- validate_specs(viz, verbose = FALSE)
if (!result) {
  print(attr(result, "issues"))
}

## End(Not run)
```

---

viz_bar *Create Bar Chart*

Description

Creates horizontal or vertical bar charts showing counts, percentages, or means. Supports simple bars or grouped bars (when `group_var` is provided). Can display error bars (standard deviation, standard error, or confidence intervals) when showing means via `value_var`.

Usage

```
viz_bar(
  data,
  x_var,
  group_var = NULL,
  value_var = NULL,
  title = NULL,
  subtitle = NULL,
  x_label = NULL,
  y_label = NULL,
  horizontal = FALSE,
  bar_type = "count",
  color_palette = NULL,
  group_order = NULL,
  x_order = NULL,
  sort_by_value = FALSE,
  sort_desc = TRUE,
  x_breaks = NULL,
  x_bin_labels = NULL,
  include_na = FALSE,
  na_label = "(Missing)",
  weight_var = NULL,
  error_bars = "none",
  ci_level = 0.95,
  error_bar_color = "black",
  error_bar_width = 50,
  tooltip = NULL,
  tooltip_prefix = "",
  tooltip_suffix = "",
  x_tooltip_suffix = "",
  data_labels_enabled = TRUE,
  label_decimals = NULL,
  legend_position = NULL,
  complete_groups = TRUE,
  y_var = NULL,
  cross_tab_filter_vars = NULL,
  title_map = NULL,
  backend = "highcharter"
)
```

Arguments

data	A data frame containing the survey data.
x_var	Character string. Name of the categorical variable for the x-axis.
group_var	Optional character string. Name of grouping variable to create separate bars (e.g., score ranges, categories). Creates grouped/clustered bars.
value_var	Optional character string. Name of a numeric variable to aggregate. When

	provided, bars show the mean of this variable per category (instead of counts). Required for error bars with "sd", "se", or "ci".
`title`	Optional main title for the chart.
`subtitle`	Optional subtitle for the chart.
`x_label`	Optional label for the x-axis. Defaults to `x_var` name.
`y_label`	Optional label for the y-axis.
`horizontal`	Logical. If `TRUE`, creates horizontal bars. Defaults to `FALSE`.
`bar_type`	Character string. Type of bar chart: "count", "percent", or "mean". Defaults to "count". When `value_var` is provided, automatically switches to "mean".
`color_palette`	Optional character vector of colors for the bars.
`group_order`	Optional character vector specifying the order of groups (for `group_var`).
`x_order`	Optional character vector specifying the order of x categories.
`sort_by_value`	Logical. If `TRUE`, sort categories by their value (highest on top for horizontal bars).
`sort_desc`	Logical. If `sort_by_value = TRUE`, sort descending (default) or ascending.
`x_breaks`	Optional numeric vector for binning continuous x variables.
`x_bin_labels`	Optional character vector of labels for x bins.
`include_na`	Logical. Whether to include NA values as a separate category. Defaults to `FALSE`.
`na_label`	Character string. Label for NA category if `include_na = TRUE`. Defaults to "(Missing)".
`weight_var`	Optional character string. Name of a weight variable to use for weighted aggregation. When provided, counts are computed as the sum of weights instead of simple counts.
`error_bars`	Character string. Type of error bars to display: "none" (default), "sd" (standard deviation), "se" (standard error), or "ci" (confidence interval). Requires `value_var` to be specified.
`ci_level`	Numeric. Confidence level for confidence intervals. Defaults to 0.95 (95% CI). Only used when `error_bars = "ci"`.
`error_bar_color`	Character string. Color for error bars. Defaults to "black".
`error_bar_width`	Numeric. Width of error bar whiskers as percentage (0-100). Defaults to 50.
`tooltip`	A tooltip configuration created with `tooltip()`, OR a format string with {placeholders}. Available placeholders: `{category}`, `{value}`, `{percent}`, `{series}`. For simple cases, use `tooltip_prefix` and `tooltip_suffix` instead. See `tooltip` for full customization options.
`tooltip_prefix`	Optional string prepended to tooltip values (simple customization).
`tooltip_suffix`	Optional string appended to tooltip values (simple customization).
`x_tooltip_suffix`	Optional string appended to x-axis category in tooltips.

data_labels_enabled
: Logical. If TRUE, show value labels on bars. Default TRUE.

label_decimals
: Optional integer. Number of decimal places for data labels. When NULL (default), uses smart defaults: 0 for counts/percent, 1 for means. Set explicitly to override (e.g., `label_decimals = 2` for two decimal places).

legend_position
: Position of the legend ("top", "bottom", "left", "right", "none")

complete_groups
: Logical. When TRUE (default), ensures all x_var/group_var combinations are present in the output, filling missing combinations with 0. This prevents bar misalignment when some groups have no observations for certain categories. Set to FALSE to show only observed combinations. Only applies when `group_var` is specified.

y_var
: Optional character string. Name of a column containing pre-aggregated counts or values. When provided, skips aggregation and uses these values directly. Useful when working with already-aggregated data (e.g., Column 1: Group, Column 2: Count).

cross_tab_filter_vars
: Optional character vector of variable names to use as cross-tab filter controls for this chart. Enables dynamic filtering.

title_map
: Optional named list mapping filter values to custom chart titles. When cross_tab_filter_vars are used, this controls the displayed title for each filter value.

backend
: Rendering backend: "highcharter" (default), "plotly", "echarts4r", or "ggiraph".

Value

A highcharter plot object.

Examples

```
## Not run:
# Simple bar chart showing counts (default)
plot1 <- viz_bar(
  data = survey_data,
  x_var = "category"
)
plot1

# Horizontal bars with percentages
plot2 <- viz_bar(
  data = survey_data,
  x_var = "category",
  horizontal = TRUE,
  bar_type = "percent"
)
plot2

# Grouped bars
plot3 <- viz_bar(
```

```
  data = survey_data,
  x_var = "question",
  group_var = "score_range",
  color_palette = c("#D2691E", "#4682B4", "#228B22"),
  group_order = c("Low (1-9)", "Middle (10-19)", "High (20-29)")
)
plot3

## End(Not run)

# Bar chart with means and error bars (95% CI)
plot4 <- viz_bar(
  data = mtcars,
  x_var = "cyl",
  value_var = "mpg",
  error_bars = "ci",
  title = "Mean MPG by Cylinders",
  y_label = "Miles per Gallon"
)
plot4

# Grouped means with standard error bars
plot5 <- viz_bar(
  data = mtcars,
  x_var = "cyl",
  group_var = "am",
  value_var = "mpg",
  error_bars = "se",
  title = "Mean MPG by Cylinders and Transmission"
)
plot5
```

`viz_boxplot` *Create a Box Plot*

Description

Creates an interactive box plot using highcharter. Supports grouped boxplots, horizontal orientation, outlier display, and weighted percentile calculations.

Usage

```
viz_boxplot(
  data,
  y_var,
  x_var = NULL,
  title = NULL,
  subtitle = NULL,
  x_label = NULL,
```

```
  y_label = NULL,
  color_palette = NULL,
  show_outliers = TRUE,
  horizontal = FALSE,
  weight_var = NULL,
  x_order = NULL,
  x_map_values = NULL,
  include_na = FALSE,
  na_label = "(Missing)",
  tooltip = NULL,
  tooltip_prefix = "",
  tooltip_suffix = "",
  legend_position = NULL,
  backend = "highcharter",
  cross_tab_filter_vars = NULL
)
```

Arguments

data	A data frame containing the variable to plot.
y_var	String. Name of the numeric column for the boxplot.
x_var	Optional string. Name of a grouping variable for multiple boxes.
title	Optional string. Main chart title.
subtitle	Optional string. Chart subtitle.
x_label	Optional string. X-axis label. Defaults to x_var.
y_label	Optional string. Y-axis label. Defaults to y_var.
color_palette	Optional character vector of colors for the boxes.
show_outliers	Logical. If TRUE, show outlier points. Default TRUE.
horizontal	Logical. If TRUE, flip chart orientation. Default FALSE.
weight_var	Optional string. Name of a weight variable for weighted percentile calculations.
x_order	Optional character vector specifying the order of x categories.
x_map_values	Optional named list to recode x_var values (e.g., list("1" = "Male", "2" = "Female")).
include_na	Logical. If TRUE, include NA groups as explicit category. Default FALSE.
na_label	String. Label for NA group when include_na = TRUE. Default "(Missing)".
tooltip	A tooltip configuration created with tooltip(), OR a format string with {placeholders}. Boxplot-specific placeholders: {category}, {high}, {q3}, {median}, {q1}, {low}. See tooltip for full customization options.
tooltip_prefix	Optional string prepended to values in tooltip.
tooltip_suffix	Optional string appended to values in tooltip.
legend_position	Position of the legend ("top", "bottom", "left", "right", "none")
backend	Rendering backend: "highcharter" (default), "plotly", "echarts4r", or "ggiraph".

cross_tab_filter_vars
: Optional character vector of variable names to use for client-side cross-tab filtering when dashboard inputs are present.

Value

A highcharter boxplot object.

Examples

```
## Not run:
# Basic boxplot
data(gss_panel20)

# Example 1: Simple boxplot of age
plot1 <- viz_boxplot(
  data = gss_panel20,
  y_var = "age",
  title = "Age Distribution"
)
plot1

# Example 2: Boxplot by education level
plot2 <- viz_boxplot(
  data = gss_panel20,
  y_var = "age",
  x_var = "degree",
  title = "Age Distribution by Education",
  x_label = "Highest Degree",
  y_label = "Age (years)"
)
plot2

# Example 3: Horizontal boxplot without outliers
plot3 <- viz_boxplot(
  data = gss_panel20,
  y_var = "age",
  x_var = "sex",
  title = "Age by Sex",
  horizontal = TRUE,
  show_outliers = FALSE
)
plot3

## End(Not run)
```

---

viz_density *Create a Density Plot*

Description

Creates an interactive kernel density estimate plot using highcharter. Supports grouped densities, weighted kernel density estimation, and customization options.

Usage

```
viz_density(
  data,
  x_var,
  group_var = NULL,
  title = NULL,
  subtitle = NULL,
  x_label = NULL,
  y_label = NULL,
  color_palette = NULL,
  fill_opacity = 0.3,
  show_rug = FALSE,
  bandwidth = NULL,
  weight_var = NULL,
  group_order = NULL,
  include_na = FALSE,
  na_label = "(Missing)",
  tooltip = NULL,
  tooltip_suffix = "",
  legend_position = NULL,
  backend = "highcharter"
)
```

Arguments

data	A data frame containing the variable to plot.
x_var	String. Name of the numeric column for density estimation.
group_var	Optional string. Name of a grouping variable for multiple overlaid densities.
title	Optional string. Main chart title.
subtitle	Optional string. Chart subtitle.
x_label	Optional string. X-axis label. Defaults to x_var.
y_label	Optional string. Y-axis label. Defaults to "Density".
color_palette	Optional character vector of colors for the density curves.
fill_opacity	Numeric between 0 and 1. Fill transparency. Default 0.3.
show_rug	Logical. If TRUE, show rug marks at the bottom. Default FALSE.
bandwidth	Optional numeric. Kernel bandwidth. If NULL (default), uses R's default bandwidth selection.
weight_var	Optional string. Name of a weight variable for weighted density estimation.
group_order	Optional character vector specifying the order of groups.

include_na	Logical. If TRUE, include NA groups as explicit category. Default FALSE.
na_label	String. Label for NA group when include_na = TRUE. Default "(Missing)".
tooltip	A tooltip configuration created with tooltip(), OR a format string with {placeholders}. Available placeholders: {x}, {y}, {value}, {series}. See tooltip for full customization options.
tooltip_suffix	Optional string appended to density values in tooltip (simple customization).
legend_position	Position of the legend ("top", "bottom", "left", "right", "none")
backend	Rendering backend: "highcharter" (default), "plotly", "echarts4r", or "ggiraph".

Value

A highcharter density plot object.

Examples

```
## Not run:
# Basic density plot
data(gss_panel20)

# Example 1: Simple density of age
plot1 <- viz_density(
  data = gss_panel20,
  x_var = "age",
  title = "Age Distribution",
  x_label = "Age (years)"
)
plot1

# Example 2: Grouped densities by sex
plot2 <- viz_density(
  data = gss_panel20,
  x_var = "age",
  group_var = "sex",
  title = "Age Distribution by Sex",
  x_label = "Age (years)",
  color_palette = c("#3498DB", "#E74C3C")
)
plot2

# Example 3: Customized density with rug marks
plot3 <- viz_density(
  data = gss_panel20,
  x_var = "age",
  title = "Age Distribution",
  fill_opacity = 0.5,
  show_rug = TRUE,
  bandwidth = 3
)
plot3
```

```
## End(Not run)
```

`viz_dumbbell` *Create a Dumbbell Chart*

Description

Creates an interactive dumbbell (dot plot range) chart using highcharter. Dumbbell charts show the difference between two values per category, useful for before/after comparisons, ranges, or gaps.

Usage

```
viz_dumbbell(
  data,
  x_var,
  low_var,
  high_var,
  title = NULL,
  subtitle = NULL,
  x_label = NULL,
  y_label = NULL,
  horizontal = TRUE,
  low_label = "Low",
  high_label = "High",
  low_color = "#E15759",
  high_color = "#4E79A7",
  connector_color = "#999999",
  connector_width = 2,
  dot_size = 6,
  x_order = NULL,
  sort_by_gap = FALSE,
  sort_desc = TRUE,
  data_labels_enabled = FALSE,
  color_palette = NULL,
  tooltip = NULL,
  tooltip_prefix = "",
  tooltip_suffix = "",
  backend = "highcharter"
)
```

Arguments

`data`	A data frame containing the data.
`x_var`	Character string. Name of the categorical variable (category labels).
`low_var`	Character string. Name of the numeric column for the lower value.

high_var	Character string. Name of the numeric column for the higher value.
title	Optional main title for the chart.
subtitle	Optional subtitle for the chart.
x_label	Optional label for the category axis.
y_label	Optional label for the value axis.
horizontal	Logical. If TRUE (default), creates horizontal dumbbells.
low_label	Character string. Label for the low-value series. Default "Low".
high_label	Character string. Label for the high-value series. Default "High".
low_color	Character string. Color for the low-value dots. Default "#E15759".
high_color	Character string. Color for the high-value dots. Default "#4E79A7".
connector_color	Character string. Color for the connecting line. Default "#999999".
connector_width	Numeric. Width of the connecting line in pixels. Default 2.
dot_size	Numeric. Radius of the dots in pixels. Default 6.
x_order	Optional character vector specifying the order of categories.
sort_by_gap	Logical. If TRUE, sort by the gap between high and low values. Default FALSE.
sort_desc	Logical. Sort direction. Default TRUE (largest gap first).
data_labels_enabled	Logical. If TRUE, show value labels. Default FALSE.
color_palette	Optional named vector of two colors: c(low = "...", high = "...").
tooltip	A tooltip configuration created with tooltip(), OR a format string with {placeholders}.
tooltip_prefix	Optional string prepended to tooltip values.
tooltip_suffix	Optional string appended to tooltip values.
backend	Rendering backend: "highcharter" (default), "plotly", "echarts4r", or "ggiraph".

Value

A highcharter plot object.

Examples

```
## Not run:
df <- data.frame(
  country = c("US", "UK", "DE", "FR"),
  score_2020 = c(65, 58, 72, 60),
  score_2024 = c(78, 65, 75, 70)
)
viz_dumbbell(df, x_var = "country",
             low_var = "score_2020", high_var = "score_2024",
             low_label = "2020", high_label = "2024",
             title = "Score Changes 2020-2024")

## End(Not run)
```

viz_funnel *Create a Funnel Chart*

Description

Creates an interactive funnel chart using highcharter. Funnel charts show sequential stages in a process, with each stage narrower than the previous, representing drop-off or conversion rates.

Usage

```
viz_funnel(
  data,
  x_var,
  y_var,
  title = NULL,
  subtitle = NULL,
  color_palette = NULL,
  x_order = NULL,
  neck_width = "30%",
  neck_height = "25%",
  show_conversion = TRUE,
  data_labels_enabled = TRUE,
  data_labels_format = "{point.name}: {point.y:,.0f}",
  show_in_legend = FALSE,
  reversed = FALSE,
  weight_var = NULL,
  tooltip = NULL,
  tooltip_prefix = "",
  tooltip_suffix = "",
  height = 400,
  backend = "highcharter"
)
```

Arguments

data	A data frame containing the data.
x_var	Character string. Name of the categorical variable (stage names).
y_var	Character string. Name of the numeric column with values per stage.
title	Optional main title for the chart.
subtitle	Optional subtitle for the chart.
color_palette	Optional character vector of colors for the stages.
x_order	Optional character vector specifying the order of stages (top to bottom). If NULL, uses the order in the data.
neck_width	Character string. Width of the funnel neck as percentage. Default "30%". Set to "0%" for a pyramid shape.

neck_height	Character string. Height of the neck section as percentage. Default "25%".
show_conversion	Logical. If TRUE (default), show conversion rates between stages in tooltips.
data_labels_enabled	Logical. If TRUE (default), show labels on stages.
data_labels_format	Character string. Format for data labels. Default "{point.name}: {point.y:,.0f}".
show_in_legend	Logical. If TRUE, show a legend. Default FALSE.
reversed	Logical. If TRUE, reverse the funnel (pyramid). Default FALSE.
weight_var	Optional character string. Name of weight variable for aggregation.
tooltip	A tooltip configuration created with `tooltip()`, OR a format string.
tooltip_prefix	Optional string prepended to tooltip values.
tooltip_suffix	Optional string appended to tooltip values.
height	Numeric. Chart height in pixels. Default 400.
backend	Rendering backend: "highcharter" (default), "plotly", "echarts4r", or "ggiraph".

Value

A highcharter plot object.

Examples

```
## Not run:
df <- data.frame(
  stage = c("Visits", "Signups", "Trial", "Purchase"),
  count = c(10000, 3000, 800, 200)
)
viz_funnel(df, x_var = "stage", y_var = "count",
           title = "Conversion Funnel")

## End(Not run)
```

---

viz_gauge *Create a Gauge or Bullet Chart*

Description

Creates an interactive gauge (speedometer) or bullet chart using highcharter. Gauges show a single value against a scale, ideal for KPIs and scores.

Usage

```
viz_gauge(
  data = NULL,
  value = NULL,
  value_var = NULL,
  min = 0,
  max = 100,
  title = NULL,
  subtitle = NULL,
  gauge_type = "solid",
  bands = NULL,
  inner_radius = "60%",
  rounded = TRUE,
  data_labels_format = "{y}",
  data_labels_style = NULL,
  color = "#4E79A7",
  background_color = "#e6e6e6",
  target = NULL,
  target_color = "#333333",
  height = 300,
  backend = "highcharter"
)
```

Arguments

data	Optional data frame. If provided, value_var is used to extract the value.
value	Numeric. The value to display on the gauge. Used when data is NULL.
value_var	Optional character string. Column name in data to use as the value. The mean/first value is extracted.
min	Numeric. Minimum value of the gauge scale. Default 0.
max	Numeric. Maximum value of the gauge scale. Default 100.
title	Optional main title for the chart.
subtitle	Optional subtitle for the chart.
gauge_type	Character string. Type of gauge: "solid" (default) or "activity". "solid" creates a solid gauge (filled arc). "activity" creates an activity-style gauge.
bands	Optional list of band definitions for color zones. Each band is a list with: from, to, color, and optionally label. Example: list(list(from = 0, to = 50, color = "red"), list(from = 50, to = 100, color = "green")).
inner_radius	Character string. Inner radius of the gauge arc as percentage. Default "60%". Increase for thinner arc, decrease for thicker.
rounded	Logical. If TRUE (default), use rounded ends on the gauge arc.
data_labels_format	Character string. Format for the center label. Default "{y}". Use "{y}%" for percentage, "${y}" for currency, etc.

data_labels_style
: Optional list of CSS styles for the center label. Default: large bold text.

color
: Character string. Color of the gauge fill. Default "#4E79A7". Ignored if bands are specified (band colors are used instead).

background_color
: Character string. Color of the gauge background track. Default "#e6e6e6".

target
: Optional numeric. Target/goal value to show as a marker on the gauge.

target_color
: Character string. Color of the target marker. Default "#333333".

height
: Numeric. Chart height in pixels. Default 300.

backend
: Rendering backend: "highcharter" (default), "plotly", "echarts4r", or "ggiraph".

Value

A highcharter plot object.

Examples

```
## Not run:
# Simple gauge
viz_gauge(value = 73, title = "Completion Rate", data_labels_format = "{y}%")

# Gauge with color bands
viz_gauge(value = 65, min = 0, max = 100,
          bands = list(
            list(from = 0, to = 40, color = "#E15759"),
            list(from = 40, to = 70, color = "#F28E2B"),
            list(from = 70, to = 100, color = "#59A14F")
          ),
          title = "Performance Score")

# From data
viz_gauge(data = mtcars, value_var = "mpg", min = 10, max = 35,
          title = "Average MPG")

## End(Not run)
```

---

viz_heatmap *Create a Heatmap*

Description

This function creates a heatmap for bivariate data, visualizing the relationship between two categorical variables and a numeric value using color intensity. It handles ordered factors, ensures all combinations are plotted, and allows for extensive customization. It also includes robust handling of missing values (NA) by allowing them to be displayed as explicit categories.

Usage

```
viz_heatmap(
  data,
  x_var,
  y_var,
  value_var,
  title = NULL,
  subtitle = NULL,
  x_label = NULL,
  y_label = NULL,
  value_label = NULL,
  tooltip = NULL,
  tooltip_prefix = "",
  tooltip_suffix = "",
  x_tooltip_suffix = "",
  y_tooltip_suffix = "",
  x_tooltip_prefix = "",
  y_tooltip_prefix = "",
  x_order = NULL,
  y_order = NULL,
  x_order_by = NULL,
  y_order_by = NULL,
  color_min = NULL,
  color_max = NULL,
  color_palette = c("#FFFFFF", "#7CB5EC"),
  na_color = "transparent",
  data_labels_enabled = TRUE,
  label_decimals = 1,
  tooltip_labels_format = NULL,
  include_na = FALSE,
  na_label_x = "(Missing)",
  na_label_y = "(Missing)",
  x_map_values = NULL,
  y_map_values = NULL,
  agg_fun = mean,
  weight_var = NULL,
  pre_aggregated = FALSE,
  legend_position = NULL,
  backend = "highcharter"
)
```

Arguments

data	A data frame containing the variables to plot.
x_var	String. Name of the column for the X-axis categories.
y_var	String. Name of the column for the Y-axis categories.
value_var	String. Name of the numeric column whose values will determine the color intensity.

`title` Optional string. Main chart title.

`subtitle` Optional string. Chart subtitle.

`x_label` Optional string. X-axis label. Defaults to `x_var`.

`y_label` Optional string. Y-axis label. Defaults to `y_var`.

`value_label` Optional string. Label for the color axis. Defaults to `value_var`.

`tooltip` A tooltip configuration created with `tooltip()`, OR a format string with {placeholders}. Available placeholders: `{x}`, `{y}`, `{value}`, `{name}`. See `tooltip` for full customization options.

`tooltip_prefix` Optional string prepended in the tooltip value (simple customization).

`tooltip_suffix` Optional string appended in the tooltip value (simple customization).

`x_tooltip_suffix`
Optional string appended to x value in tooltip.

`y_tooltip_suffix`
Optional string appended to y value in tooltip.

`x_tooltip_prefix`
Optional string prepended to x value in tooltip.

`y_tooltip_prefix`
Optional string prepended to y value in tooltip.

`x_order` Optional character vector to order the factor levels of `x_var`. Alternatively, use `x_order_by` to order by aggregated values.

`y_order` Optional character vector to order the factor levels of `y_var`. Alternatively, use `y_order_by` to order by aggregated values.

`x_order_by` Optional. Order x-axis categories by aggregated value. Can be "asc" (ascending), "desc" (descending), or NULL (default, no reordering). When set, categories are sorted by their mean value across all y categories.

`y_order_by` Optional. Order y-axis categories by aggregated value. Can be "asc" (ascending), "desc" (descending), or NULL (default, no reordering). When set, categories are sorted by their mean value across all x categories.

`color_min` Optional numeric. Minimum value for the color axis. If NULL, defaults to data min.

`color_max` Optional numeric. Maximum value for the color axis. If NULL, defaults to data max.

`color_palette` Optional character vector of colors for the color gradient. Example: `c("#FFFFFF", "#7CB5EC")` for white to light blue. Can also be a single color for gradient start.

`na_color` Optional string. Color for NA values in `value_var` cells. Default "transparent".

`data_labels_enabled`
Logical. If TRUE, display data labels on each cell. Default TRUE.

`label_decimals` Integer. Number of decimal places for data labels and tooltips. Default is 1. Set to 0 for whole numbers, 2 for two decimal places, etc. Ignored if `tooltip_labels_format` is explicitly provided.

`tooltip_labels_format`
Optional string. Format for data labels. Default NULL (auto-generated from `label_decimals`). If provided, overrides `label_decimals`.

include_na	Logical. If TRUE, treats NA values in x_var or y_var as explicit categories using na_label_x and na_label_y. If FALSE (default), rows with NA in x_var or y_var are excluded from aggregation.
na_label_x	Optional string. Custom label for NA values in x_var. Defaults to "(Missing)".
na_label_y	Optional string. Custom label for NA values in y_var. Defaults to "(Missing)".
x_map_values	Optional named list to recode x_var values for display.
y_map_values	Optional named list to recode y_var values for display.
agg_fun	Function to aggregate duplicate x/y combinations. Default is mean. Note: If weight_var is provided, weighted mean is used instead and this parameter is ignored.
weight_var	Optional string. Name of a weight variable to use for weighted mean aggregation. When provided, the function uses weighted.mean() instead of the agg_fun parameter.
pre_aggregated	Logical. If TRUE, skips aggregation and uses value_var directly. Use this when your data is already aggregated (one row per x/y combination). Default is FALSE.
legend_position	Position of the legend ("top", "bottom", "left", "right", "none")
backend	Rendering backend: "highcharter" (default), "plotly", "echarts4r", or "ggiraph".

Details

This function performs the following steps:

1. **Input validation:** Checks data, x_var, y_var, and value_var.
2. **Data Preparation:**
   - Handles haven_labelled columns by converting them to factors.
   - Applies value mapping if x_map_values or y_map_values (new parameter) are provided.
   - Processes NA values in x_var and y_var: if include_na = TRUE, NAs are converted to a specified label; otherwise, rows with NAs in these variables are filtered out.
   - Converts x_var and y_var to factors and applies x_order and y_order.
   - If weight_var is provided, uses weighted.mean() for aggregation; otherwise uses agg_fun (default mean()).
   - Uses tidyr::complete to ensure all x_var/y_var combinations are present, filling missing value_var with NA_real_ (which will appear as na_color in the heatmap).
3. **Chart Construction:**
   - Initializes a highchart object.
   - Configures title, subtitle, axis labels.
   - Sets up hc_colorAxis based on color_min, color_max, and color_palette.
   - Adds the heatmap series using hc_add_series, mapping x_var, y_var, and value_var.
   - Customizes plotOptions for heatmap, including data labels and nullColor.
4. **Tooltip Customization:** Defines a JavaScript tooltip.formatter for detailed hover information.

Value

A highcharter heatmap object.

Examples

```
## Not run:
# Load the dataset
data(gss_panel20)

# Example 1: Basic heatmap - no mapped values or other customization
viz_heatmap(
  data = gss_panel20,
  x_var = "degree_1a",
  y_var = "sex_1a",
  value_var = "age_1a",
  title = "Average Age by Education and Sex",
  x_label = "Education Level",
  y_label = "Sex",
  value_label = "Mean Age"
)


# Example 2: Heatmap With Custom Variable Mapping and Colors

region_map <- list("1" = "New England",
"2" = "Mid-Atlantic",
"3" = "East North Central",
"4" = "West North Central",
"5" = "South Atlantic",
"6" = "Deep South",
"7" = "West South Central",
"8" = "Mountain",
"9" = "West Coast"
)
sex_map <- list("1" = "Male",
                "2" = "Female")

viz_heatmap(
  data = gss_panel20,
  x_var = "region_1a",
  y_var = "sex_1a",
  value_var = "satfin_1a",
  x_map_values = region_map,
  y_map_values = sex_map,
  value_label = "Satisfaction",
  x_label = "U.S. Region",
  y_label = "Gender",
  title = "Satisfaction with Financial Situation",
  subtitle = "Per U.S. Region and Gender",
  color_palette = c("#f7fbff", "darkgreen"),
  color_min = 1,
  color_max = 3
```

```
)


# Example 3: Handling missing categories explicitly

edu_map = list("0" = "less than high school",
"1" =  "high school",
"2" = "associate/junior college",
"3" = "bachelor's",
"4" = "graduate")

viz_heatmap(
data = gss_panel20,
x_var = "region_1a",
y_var = "degree_1a",
value_var = "income_1a",
x_map_values = region_map,
y_map_values = edu_map,
color_min = 8,
color_max = 12,
value_label = "Income",
x_label = "U.S. Region",
y_label = "Education",
include_na = TRUE,
na_label_x = "Region Missing",
na_label_y = "Degree Missing",
na_color = "grey",
title = "Average Income by Region and Education (Including Missing)"
)


# Example 4: Custom order of education levels and relabeling of sex
viz_heatmap(
data = gss_panel20,
x_var = "degree_1a",
y_var = "sex_1a",
value_var = "income_1a",
x_map_values = edu_map,
x_order = c("less than high school", "high school", "associate/junior college",
"bachelor's", "graduate"),
y_map_values = sex_map,
y_label = "Gender",
x_label = "Education Level",
value_label = "Income Level",
title = "Average Income by Education Level and Sex",
subtitle = "Custom order and relabeled categories",
color_palette = c("#ffffe0", "#31a354")
)

## End(Not run)
```

viz_histogram *Create an Histogram*

Description

This function creates a histogram for survey data. It handles raw (unaggregated) data, counting the occurences of categories, supporting ordered factors, allowing numerical x-axis variables to be binned into custom groups, and enables renaming of categorical values for display. It can also handle SPSS (.sav) columns automatically.

Usage

```
viz_histogram(
  data,
  x_var,
  y_var = NULL,
  title = NULL,
  subtitle = NULL,
  x_label = NULL,
  y_label = NULL,
  histogram_type = c("count", "percent"),
  tooltip = NULL,
  tooltip_prefix = "",
  tooltip_suffix = "",
  x_tooltip_suffix = "",
  bins = NULL,
  bin_breaks = NULL,
  bin_labels = NULL,
  include_na = FALSE,
  na_label = "(Missing)",
  color_palette = NULL,
  x_map_values = NULL,
  x_order = NULL,
  weight_var = NULL,
  data_labels_enabled = TRUE,
  label_decimals = NULL,
  legend_position = NULL,
  backend = "highcharter"
)
```

Arguments

data	A data frame containing the variable to plot.
x_var	String. Name of the column to histogram (numeric or categorical).
y_var	Optional string. Name of a pre-computed count column. If supplied, the function skips counting and uses this column as y.

title	Optional string. Main chart title.
subtitle	Optional string. Chart subtitle.
x_label	Optional string. X-axis label. Defaults to x_var.
y_label	Optional string. Y-axis label. Defaults to "Count" or "Percentage".
histogram_type	One of "count" or "percent". Default "count".
tooltip	A tooltip configuration created with tooltip(), OR a format string with {placeholders}. Available placeholders: {category}, {value}, {percent}. For simple cases, use tooltip_prefix and tooltip_suffix instead. See tooltip for full customization options.
tooltip_prefix	Optional string prepended in the tooltip (simple customization).
tooltip_suffix	Optional string appended in the tooltip (simple customization).
x_tooltip_suffix	Optional string appended to x value in tooltip.
bins	Optional integer. Number of bins to compute via hist().
bin_breaks	Optional numeric vector of cut points.
bin_labels	Optional character vector of labels for the bins. Must be length length(breaks)-1.
include_na	Logical. If TRUE, NA values are shown as explicit categories in the visualization. If FALSE (default), rows with NA in the x variable are excluded. Default FALSE.
na_label	String. Label to display for NA values when include_na = TRUE. Default "(Missing)".
color_palette	Optional string or vector of colors for the bars.
x_map_values	Optional named list to recode raw x_var values before binning (e.g., list("1" = "Male", "2" = "Female")).
x_order	Optional character vector to order the factor levels of the binned variable.
weight_var	Optional string. Name of a weight variable to use for weighted aggregation. When provided, counts are computed as the sum of weights instead of simple counts.
data_labels_enabled	Logical. If TRUE, show value labels on bars. Default TRUE.
label_decimals	Optional integer. Number of decimal places for data labels. When NULL (default), uses smart defaults: 0 for counts, 1 for percent. Set explicitly to override (e.g., label_decimals = 2).
legend_position	Position of the legend ("top", "bottom", "left", "right", "none")
backend	Rendering backend: "highcharter" (default), "plotly", "echarts4r", or "ggiraph".

Details

This function performs the following steps:

1. **Input validation:** Checks that data is a data frame and x_var (and y_var if given) exist.
2. **Haven-labelled handling:** If x_var is of class "haven_labelled", converts it to numeric.

3. **Value mapping:** If `x_map_values` is provided, recodes raw values before any binning.
4. **Binning:**
   - If `bins` is set (and `bin_breaks` is `NULL`), computes breaks via `hist()`.
   - If `bin_breaks` is provided, cuts `x_var` into categories, using `bin_labels` if supplied.
   - Otherwise uses the raw `x_var` values.
5. **Factor and NA handling:** Converts the plotting variable to a factor; if `include_na = TRUE`, adds an explicit "(NA)" level. Applies `x_order` if given.
6. **Aggregation:**
   - If `y_var` is `NULL`, counts occurrences of each factor level.
   - Otherwise renames `y_var` to `n` and skips counting.
7. **Chart construction:** Builds a `highcharter` column chart of `n` vs. the factor levels.
8. **Customization:**
   - Applies `title`, `subtitle`, axis labels.
   - Sets stacking mode (for percent vs. count), data labels format.
   - Defines a JS `tooltip.formatter` using `tooltip_prefix`, `tooltip_suffix`, and `x_tooltip_suffix`.
   - Applies custom `color` if provided.

Value

A `highcharter` histogram (column) plot object.

Examples

```
## Not run:
# We will work with data from the GSS.
data(gss_panel20)

# Example 1: Basic histogram of age distribution
plot1 <- viz_histogram(
  data = gss_panel20,
  x_var = "age",
  title = "Age Distribution in GSS Data (2010+)",
  subtitle = "General Social Survey respondents",
  x_label = "Age (years)",
  y_label = "Number of Respondents",
  bins = 15,
  color_palette = "steelblue"
)
plot1

# Example 2: Education levels with custom labels (excluding NAs)
education_map <- list("0" = "Less than High School",
                      "1" = "High School",
                      "2" = "Associate/Junior College",
                      "3" = "Bachelor's",
                      "4" = "Graduate"
                      )
```

```
plot2 <- viz_histogram(
 data = gss_panel20,
 x_var = "degree",
 title = "Educational Attainment Distribution",
 subtitle = "GSS respondents 2010-present (NAs excluded)",
 x_label = "Highest Degree Completed",
 y_label = "Count",
 histogram_type = "count",
 x_map_values = education_map,
 color_palette = "pink",
 include_na = FALSE  # Exclude missing values
)
plot2

# Example 3: Including NA values with custom label
plot3 <- viz_histogram(
  data = gss_panel20,
  x_var = "degree",
  title = "Educational Attainment Distribution (Including Missing Data)",
  subtitle = "GSS respondents 2010-present",
  x_label = "Highest Degree Completed",
  x_order = education_order,
  include_na = TRUE,  # Show NAs as explicit category
  na_label = "Not Reported"  # Custom label for NAs
)
plot3

# Example 4: Age binning with custom breaks
age_breaks <- c(18, 30, 45, 60, 75, Inf)
age_labels <- c("18-29", "30-44", "45-59", "60-74", "75+")

plot5 <- viz_histogram(
  data = gss_panel20,
  x_var = "age",
  title = "Age Groups in GSS Sample",
  subtitle = "Custom age categories",
  x_label = "Age Group",
  y_label = "Number of Respondents",
  bin_breaks = age_breaks,
  bin_labels = age_labels,
  tooltip_prefix = "Count: ",
  x_tooltip_suffix = " years old",
  color_palette = "seagreen"
)
plot5

## End(Not run)
```

---

viz_lollipop *Create a Lollipop Chart*

Description

Creates an interactive lollipop chart using highcharter. A lollipop chart is a bar chart variant that uses a line (stem) and dot instead of a full bar, making it easier to read when there are many categories.

Usage

```
viz_lollipop(
  data,
  x_var,
  y_var = NULL,
  group_var = NULL,
  value_var = NULL,
  title = NULL,
  subtitle = NULL,
  x_label = NULL,
  y_label = NULL,
  horizontal = TRUE,
  bar_type = "count",
  color_palette = NULL,
  x_order = NULL,
  group_order = NULL,
  sort_by_value = FALSE,
  sort_desc = TRUE,
  weight_var = NULL,
  dot_size = 8,
  stem_width = 2,
  data_labels_enabled = TRUE,
  label_decimals = NULL,
  tooltip = NULL,
  tooltip_prefix = "",
  tooltip_suffix = "",
  backend = "highcharter"
)
```

Arguments

data	A data frame containing the data.
x_var	Character string. Name of the categorical variable for the axis.
y_var	Optional character string. Name of a numeric column with pre-aggregated values. When provided, skips counting and uses these values directly.
group_var	Optional character string. Name of grouping variable for separate series (creates multiple dots per category).
value_var	Optional character string. Name of a numeric variable to aggregate (shows mean per category instead of counts).
title	Optional main title for the chart.
subtitle	Optional subtitle for the chart.

x_label	Optional label for the category axis.
y_label	Optional label for the value axis.
horizontal	Logical. If TRUE (default), creates horizontal lollipops.
bar_type	Character string. "count" (default), "percent", or "mean".
color_palette	Optional character vector of colors.
x_order	Optional character vector specifying the order of categories.
group_order	Optional character vector specifying the order of groups.
sort_by_value	Logical. If TRUE, sort categories by value. Default FALSE.
sort_desc	Logical. Sort direction when sort_by_value = TRUE. Default TRUE.
weight_var	Optional character string. Name of weight variable.
dot_size	Numeric. Size of the dots in pixels. Default 8.
stem_width	Numeric. Width of the stem lines in pixels. Default 2.
data_labels_enabled	Logical. If TRUE (default), show value labels.
label_decimals	Optional integer. Number of decimal places for data labels. When NULL (default), uses smart defaults: 0 for counts, 1 for percent. Set explicitly to override (e.g., label_decimals = 2).
tooltip	A tooltip configuration created with tooltip(), OR a format string with {placeholders}.
tooltip_prefix	Optional string prepended to tooltip values.
tooltip_suffix	Optional string appended to tooltip values.
backend	Rendering backend: "highcharter" (default), "plotly", "echarts4r", or "ggiraph".

Value

A highcharter plot object.

Examples

```
## Not run:
# Simple lollipop chart
viz_lollipop(mtcars, x_var = "cyl", title = "Cars by Cylinders")

# Horizontal with pre-aggregated data
df <- data.frame(country = c("US", "UK", "DE"), score = c(85, 72, 68))
viz_lollipop(df, x_var = "country", y_var = "score", horizontal = TRUE)

## End(Not run)
```

viz_map *Create an interactive map visualization*

Description

Creates a Highcharter choropleth map with optional click navigation to other pages/dashboards. Designed to work within the dashboardr visualization system via add_viz(type = "map", ...).

Usage

```
viz_map(
  data,
  map_type = "custom/world",
  value_var,
  join_var = "iso2c",
  title = NULL,
  subtitle = NULL,
  legend_title = NULL,
  color_stops = NULL,
  color_palette = c("#f7fbff", "#08306b"),
  na_color = "#E0E0E0",
  click_url_template = NULL,
  click_var = NULL,
  tooltip = NULL,
  tooltip_vars = NULL,
  tooltip_format = NULL,
  height = 500,
  border_color = "#FFFFFF",
  border_width = 0.5,
  credits = FALSE,
  backend = "highcharter",
  ...
)
```

Arguments

data	Data frame with geographic data
map_type	Map type: "custom/world", "custom/world-highres", "countries/us/us-all", etc. See Highcharts map collection.
value_var	Column name for color values (required)
join_var	Column name to join with map geography (default: "iso2c")
title	Chart title
subtitle	Chart subtitle
legend_title	Legend title (defaults to value_var)
color_stops	Numeric vector of color gradient stops

color_palette	Character vector of colors for gradient
na_color	Color for missing/NA values (default: "#E0E0E0")
click_url_template	URL template for click navigation. Use {var} syntax for variable substitution, e.g., "{iso2c}_dashboard/index.html"
click_var	Variable to use in click URL (defaults to join_var)
tooltip	A tooltip configuration created with tooltip(), OR a format string with {placeholders}. Available placeholders: {name}, {value}. See tooltip for full customization options.
tooltip_vars	Character vector of variables to show in tooltip (legacy).
tooltip_format	Custom tooltip format string using Highcharts syntax (legacy). For the simpler dashboardr placeholder syntax, use tooltip instead.
height	Chart height in pixels (default: 500)
border_color	Border color between regions (default: "#FFFFFF")
border_width	Border width (default: 0.5
credits	Show Highcharts credits (default: FALSE)
backend	Rendering backend: "highcharter" (default), "plotly", "echarts4r", or "ggiraph".
...	Additional arguments passed to highcharter::hcmap()

Value

A highchart object

Examples

```
## Not run:
# Basic world map
country_data <- data.frame(
  iso2c = c("US", "DE", "FR"),
  total_ads = c(1000, 500, 300)
)

viz_map(
  data = country_data,
  value_var = "total_ads",
  join_var = "iso2c",
  title = "Ad Spending by Country"
)

# With click navigation to country dashboards
viz_map(
  data = country_data,
  value_var = "total_ads",
  click_url_template = "{iso2c}_dashboard/index.html",
  title = "Click a country to explore"
)
```

```
# Using with add_viz()
viz <- create_viz() %>%
  add_viz(
    type = "map",
    value_var = "total_ads",
    join_var = "iso2c",
    click_url_template = "{iso2c}_dashboard/index.html"
  )

## End(Not run)
```

---

viz_pie *Create a Pie or Donut Chart*

Description

Creates an interactive pie or donut chart using highcharter. Pie charts show proportional data as slices of a circle. Donut charts are pie charts with a hollow center.

Usage

```
viz_pie(
  data,
  x_var,
  y_var = NULL,
  title = NULL,
  subtitle = NULL,
  inner_size = "0%",
  color_palette = NULL,
  x_order = NULL,
  sort_by_value = FALSE,
  data_labels_enabled = TRUE,
  data_labels_format = "{point.name}: {point.percentage:.1f}%",
  show_in_legend = TRUE,
  weight_var = NULL,
  include_na = FALSE,
  na_label = "(Missing)",
  tooltip = NULL,
  tooltip_prefix = "",
  tooltip_suffix = "",
  center_text = NULL,
  legend_position = NULL,
  backend = "highcharter",
  cross_tab_filter_vars = NULL
)
```

Arguments

data	A data frame containing the data.
x_var	Character string. Name of the categorical variable (slice labels).
y_var	Optional character string. Name of a numeric column with pre-aggregated values. When provided, skips counting and uses these values directly.
title	Optional main title for the chart.
subtitle	Optional subtitle for the chart.
inner_size	Character string. Size of the inner hole as percentage (e.g., "50%"). Set to "0%" for a standard pie chart (default), or "40%"-"70%" for a donut chart.
color_palette	Optional character vector of colors for the slices.
x_order	Optional character vector specifying the order of slices.
sort_by_value	Logical. If TRUE, sort slices by value (largest first). Default FALSE.
data_labels_enabled	Logical. If TRUE (default), show labels on slices.
data_labels_format	Character string. Format for data labels. Default shows name and percentage: "{point.name}: {point.percentage:.1f}%".
show_in_legend	Logical. If TRUE (default), show a legend.
weight_var	Optional character string. Name of a weight variable for weighted counts.
include_na	Logical. Whether to include NA as a category. Default FALSE.
na_label	Character string. Label for the NA category. Default "(Missing)".
tooltip	A tooltip configuration created with `tooltip()`, OR a format string with {placeholders}. Available placeholders: `{name}`, `{value}`, `{percent}`. See `tooltip` for full customization options.
tooltip_prefix	Optional string prepended to tooltip values.
tooltip_suffix	Optional string appended to tooltip values.
center_text	Optional character string. Text to display in the center of a donut chart. Only visible when inner_size > "0%".
legend_position	Position of the legend ("top", "bottom", "left", "right", "none")
backend	Rendering backend: "highcharter" (default), "plotly", "echarts4r", or "ggiraph".
cross_tab_filter_vars	Optional character vector of variable names to use for client-side cross-tab filtering when dashboard inputs are present.

Value

A highcharter plot object.

Examples

```
## Not run:
# Simple pie chart
viz_pie(mtcars, x_var = "cyl", title = "Cars by Cylinders")

# Donut chart
viz_pie(mtcars, x_var = "cyl", inner_size = "50%", title = "Donut Chart")

# Pre-aggregated data
df <- data.frame(category = c("A", "B", "C"), count = c(40, 35, 25))
viz_pie(df, x_var = "category", y_var = "count")

## End(Not run)
```

---

viz_sankey *Create a Sankey Diagram*

Description

Creates an interactive Sankey (alluvial flow) diagram using highcharter. Sankey diagrams show flows between nodes, where the width of each link is proportional to the flow quantity.

Usage

```
viz_sankey(
  data,
  from_var,
  to_var,
  value_var,
  title = NULL,
  subtitle = NULL,
  color_palette = NULL,
  node_width = 20,
  node_padding = 10,
  link_opacity = 0.5,
  data_labels_enabled = TRUE,
  curvature = 0.33,
  tooltip = NULL,
  tooltip_prefix = "",
  tooltip_suffix = "",
  height = 400,
  backend = "highcharter"
)
```

Arguments

data	A data frame containing the flow data.

from_var	Character string. Name of the column with source node names.
to_var	Character string. Name of the column with target node names.
value_var	Character string. Name of the numeric column with flow values/weights.
title	Optional main title for the chart.
subtitle	Optional subtitle for the chart.
color_palette	Optional character vector of colors for the nodes.
node_width	Numeric. Width of the node rectangles in pixels. Default 20.
node_padding	Numeric. Vertical padding between nodes in pixels. Default 10.
link_opacity	Numeric. Opacity of the flow links (0-1). Default 0.5.
data_labels_enabled	Logical. If TRUE (default), show labels on nodes.
curvature	Numeric. Curvature factor for links (0-1). Default 0.33.
tooltip	A tooltip configuration created with tooltip(), OR a format string.
tooltip_prefix	Optional string prepended to tooltip values.
tooltip_suffix	Optional string appended to tooltip values.
height	Numeric. Chart height in pixels. Default 400.
backend	Rendering backend: "highcharter" (default), "plotly", "echarts4r", or "ggiraph".

Value

A highcharter plot object.

Examples

```
## Not run:
df <- data.frame(
  from = c("A", "A", "B", "B"),
  to = c("X", "Y", "X", "Y"),
  flow = c(30, 20, 10, 40)
)
viz_sankey(df, from_var = "from", to_var = "to", value_var = "flow",
           title = "Flow Diagram")

## End(Not run)
```

---

viz_scatter *Create Scatter Plot*

Description

Creates interactive scatter plots showing relationships between two continuous variables. Supports optional color grouping, custom sizing, and trend lines.

Usage

```
viz_scatter(
  data,
  x_var,
  y_var,
  color_var = NULL,
  size_var = NULL,
  title = NULL,
  subtitle = NULL,
  x_label = NULL,
  y_label = NULL,
  color_palette = NULL,
  point_size = 4,
  show_trend = FALSE,
  trend_method = "lm",
  alpha = 0.7,
  include_na = FALSE,
  na_label = "(Missing)",
  tooltip = NULL,
  tooltip_format = NULL,
  jitter = FALSE,
  jitter_amount = 0.2,
  legend_position = NULL,
  backend = "highcharter",
  cross_tab_filter_vars = NULL
)
```

Arguments

data	A data frame containing the data.
x_var	Character string. Name of the variable for the x-axis (continuous or categorical).
y_var	Character string. Name of the variable for the y-axis (continuous).
color_var	Optional character string. Name of grouping variable for coloring points.
size_var	Optional character string. Name of variable to control point sizes.
title	Optional main title for the chart.
subtitle	Optional subtitle for the chart.
x_label	Optional label for the x-axis. Defaults to x_var name.
y_label	Optional label for the y-axis. Defaults to y_var name.
color_palette	Optional character vector of colors for the points.
point_size	Numeric. Default size for points when size_var is not specified. Defaults to 4.
show_trend	Logical. Whether to add a trend line. Defaults to FALSE.
trend_method	Character string. Method for trend line: "lm" (linear) or "loess". Defaults to "lm".
alpha	Numeric between 0 and 1. Transparency of points. Defaults to 0.7.

include_na Logical. Whether to include NA values in color grouping. Defaults to FALSE.

na_label Character string. Label for NA category if include_na = TRUE. Defaults to "(Missing)".

tooltip A tooltip configuration created with tooltip(), OR a format string with {placeholders}. Available placeholders: {x}, {y}, {name}, {series}. See tooltip for full customization options.

tooltip_format Character string. Custom format for tooltips using Highcharts placeholders like {point.x}, {point.y}. For the simpler dashboardr placeholder syntax, use the tooltip parameter instead.

jitter Logical. Whether to add jittering to reduce overplotting. Defaults to FALSE.

jitter_amount Numeric. Amount of jittering if jitter = TRUE. Defaults to 0.2.

legend_position
Position of the legend ("top", "bottom", "left", "right", "none")

backend Rendering backend: "highcharter" (default), "plotly", "echarts4r", or "ggiraph".

cross_tab_filter_vars
Optional character vector of variable names to use for client-side cross-tab filtering when dashboard inputs are present.

Value

A highcharter plot object.

Examples

```
# Simple scatter plot
plot1 <- viz_scatter(
  data = mtcars,
  x_var = "wt",
  y_var = "mpg",
  title = "Car Weight vs MPG"
)
plot1

# Scatter plot with color grouping
plot2 <- viz_scatter(
  data = iris,
  x_var = "Sepal.Length",
  y_var = "Sepal.Width",
  color_var = "Species",
  title = "Iris Sepal Measurements"
)
plot2

# Scatter with trend line and custom colors
plot3 <- viz_scatter(
  data = mtcars,
  x_var = "hp",
  y_var = "mpg",
  color_var = "cyl",
```

```
  show_trend = TRUE,
  title = "Horsepower vs MPG by Cylinders",
  color_palette = c("#FF6B6B", "#4ECDC4", "#45B7D1")
)
plot3
```

`viz_stackedbar` *Create a Stacked Bar Chart*

Description

A unified function for creating stacked bar charts that supports two modes:

Mode 1: Grouped/Crosstab Mode (use `x_var` + `stack_var`)

Creates a stacked bar chart from long/tidy data where one column provides the x-axis categories and another column provides the stack segments. This is ideal for cross-tabulating responses by demographic groups.

Mode 2: Multi-Variable/Battery Mode (use `x_vars`)

Creates a stacked bar chart from wide data where multiple columns become the x-axis bars, and their values become the stacks. This is ideal for comparing response distributions across multiple survey questions.

The function automatically detects which mode to use based on the parameters provided.

Usage

```
viz_stackedbar(
  data,
  x_var = NULL,
  y_var = NULL,
  stack_var = NULL,
  x_vars = NULL,
  x_var_labels = NULL,
  response_levels = NULL,
  show_var_tooltip = TRUE,
  title = NULL,
  subtitle = NULL,
  x_label = NULL,
  y_label = NULL,
  stack_label = NULL,
  stacked_type = c("counts", "percent", "normal"),
  tooltip = NULL,
  tooltip_prefix = "",
  tooltip_suffix = "",
  x_tooltip_suffix = "",
  color_palette = NULL,
  stack_order = NULL,
```

```
  x_order = NULL,
  include_na = FALSE,
  na_label_x = "(Missing)",
  na_label_stack = "(Missing)",
  x_breaks = NULL,
  x_bin_labels = NULL,
  x_map_values = NULL,
  stack_breaks = NULL,
  stack_bin_labels = NULL,
  stack_map_values = NULL,
  horizontal = FALSE,
  weight_var = NULL,
  data_labels_enabled = TRUE,
  label_decimals = NULL,
  cross_tab_filter_vars = NULL,
  title_map = NULL,
  legend_position = NULL,
  backend = "highcharter"
)
```

Arguments

data	A data frame containing the survey data.
x_var	String. Name of the column for X-axis categories (Mode 1: Grouped/Crosstab). Use this together with stack_var for crosstab-style charts.
y_var	Optional string. Name of a pre-computed count column. If NULL (default), the function counts occurrences.
stack_var	String. Name of the column whose values define the stacks. Required when using x_var.
x_vars	Character vector of column names to compare (Mode 2: Multi-Variable/Battery). Each column becomes a bar on the x-axis, and the values within each column become the stacks. Use this for comparing multiple survey questions with the same response scale.
x_var_labels	Optional character vector of display labels for the variables. Must be the same length as x_vars. If NULL, column names are used.
response_levels	Optional character vector of factor levels for the response categories (e.g., c("Strongly Disagree", ..., "Strongly Agree")). This sets the order of the stacks in multi-variable mode.
show_var_tooltip	Logical. If TRUE (default), shows enhanced tooltips with variable labels in multi-variable mode.
title	Optional string. Main chart title.
subtitle	Optional string. Chart subtitle.
x_label	Optional string. X-axis label. Defaults to empty in crosstab mode or "Variable" in multi-variable mode.

y_label	Optional string. Y-axis label. Defaults to "Count" or "Percentage".
stack_label	Optional string. Title for the stack legend. Set to NULL, NA, FALSE, or "" to hide the legend title.
stacked_type	One of "normal", "counts" (both show raw counts), or "percent" (100% stacked). Defaults to "counts".
tooltip	A tooltip configuration created with tooltip(), OR a format string with {placeholders}. Available placeholders: {category}, {value}, {series}, {percent}.
tooltip_prefix	Optional string prepended to tooltip values.
tooltip_suffix	Optional string appended to tooltip values.
x_tooltip_suffix	Optional string appended to x-axis values in tooltips.
color_palette	Optional character vector of colors for the stacks.
stack_order	Optional character vector specifying order of stack levels.
x_order	Optional character vector specifying order of x-axis levels.
include_na	Logical. If TRUE, NA values are shown as explicit categories. If FALSE (default), rows with NA are excluded.
na_label_x	String. Label for NA values on x-axis. Default "(Missing)".
na_label_stack	String. Label for NA values in stacks. Default "(Missing)".
x_breaks	Optional numeric vector of cut points for binning x_var.
x_bin_labels	Optional character vector of labels for x_breaks bins.
x_map_values	Optional named list to remap x-axis values for display.
stack_breaks	Optional numeric vector of cut points for binning stack variable.
stack_bin_labels	Optional character vector of labels for stack_breaks bins.
stack_map_values	Optional named list to remap stack values for display.
horizontal	Logical. If TRUE, creates horizontal bars. Default FALSE.
weight_var	Optional string. Name of a weight variable for weighted counts.
data_labels_enabled	Logical. If TRUE, show value labels on bars. Default TRUE.
label_decimals	Optional integer. Number of decimal places for data labels. When NULL (default), uses smart defaults: 0 for counts, 1 for percent. Set explicitly to override (e.g., label_decimals = 2).
cross_tab_filter_vars	Character vector. Variables for cross-tab filtering (typically auto-detected from sidebar inputs).
title_map	Named list mapping variable names to custom display titles for dynamic title updates when filtering by cross-tab variables.
legend_position	Position of the legend ("top", "bottom", "left", "right", "none")
backend	Rendering backend: "highcharter" (default), "plotly", "echarts4r", or "ggiraph".

Details

Choosing the Right Mode:

Use **Mode 1** (`x_var` + `stack_var`) when you want to:

- Show how one variable breaks down by another (e.g., education by gender)
- Create a cross-tabulation visualization
- Your data is already in long/tidy format

Use **Mode 2** (`x_vars`) when you want to:

- Compare response distributions across multiple survey questions
- Visualize a Likert scale battery
- Your questions share the same response categories
- Your data is in wide format (one column per question)

Data Handling Features:

- Automatically handles `haven_labelled` columns from SPSS/Stata/SAS
- Supports value mapping to rename categories for display
- Supports binning of continuous variables
- Handles missing values explicitly or implicitly

Value

An interactive `highcharter` bar chart plot object.

See Also

`viz_bar` for simple (non-stacked) bar charts

Examples

```
## Not run:
library(gssr)
data(gss_panel20)

# MODE 1: Grouped/Crosstab - One variable broken down by another

# Example 1: Education by Gender (counts)
plot1 <- viz_stackedbar(
  data = gss_panel20,
  x_var = "degree_1a",
  stack_var = "sex_1a",
  title = "Educational Attainment by Gender",
  x_label = "Highest Degree",
  stack_label = "Gender"
)

# Example 2: Happiness by Education (percentages)
```

```
plot2 <- viz_stackedbar(
  data = gss_panel20,
  x_var = "degree_1a",
  stack_var = "happy_1a",
  title = "Happiness by Education Level",
  stacked_type = "percent",
  tooltip_suffix = "%"
)

# MODE 2: Multi-Variable/Battery - Compare multiple questions

# Example 3: Compare multiple attitude questions
plot3 <- viz_stackedbar(
  data = gss_panel20,
  x_vars = c("trust_1a", "fair_1a", "helpful_1a"),
  x_var_labels = c("Trust Others", "Others Are Fair",
    "Others Are Helpful"),
  title = "Social Trust Battery",
  stacked_type = "percent",
  tooltip_suffix = "%"
)

# Example 4: Single question horizontal (compact display)
plot4 <- viz_stackedbar(
  data = gss_panel20,
  x_vars = "happy_1a",
  title = "General Happiness",
  stacked_type = "percent",
  horizontal = TRUE
)

## End(Not run)
```

`viz_stackedbars` *Stacked Bar Charts for Multiple Variables (Superseded)*

Description

Note: This function has been superseded by `viz_stackedbar`, which now supports both single-variable crosstabs and multi-variable comparisons through a unified interface. This function will continue to work but we recommend using `viz_stackedbar()` for new code.

Migration: Replace `viz_stackedbars(data, x_vars = ...)` with `viz_stackedbar(data, x_vars = ...)`. All parameters work the same way.

Usage

```
viz_stackedbars(
  data,
```

```
  x_vars,
  x_var_labels = NULL,
  response_levels = NULL,
  title = NULL,
  subtitle = NULL,
  x_label = NULL,
  y_label = NULL,
  stack_label = NULL,
  stacked_type = c("normal", "percent", "counts"),
  tooltip = NULL,
  tooltip_prefix = "",
  tooltip_suffix = "",
  x_tooltip_suffix = "",
  color_palette = NULL,
  stack_order = NULL,
  x_order = NULL,
  include_na = FALSE,
  na_label_x = "(Missing)",
  na_label_stack = "(Missing)",
  x_breaks = NULL,
  x_bin_labels = NULL,
  x_map_values = NULL,
  stack_breaks = NULL,
  stack_bin_labels = NULL,
  stack_map_values = NULL,
  show_var_tooltip = TRUE,
  horizontal = FALSE,
  weight_var = NULL,
  data_labels_enabled = TRUE
)
```

Arguments

`data`	A data frame containing the survey data.
`x_vars`	Character vector of column names to pivot (the variables to compare).
`x_var_labels`	Optional character vector of display labels for the variables. Must be the same length as `x_vars`. If `NULL`, column names are used as labels.
`response_levels`	Optional character vector of factor levels for the response categories (e.g. `c("Strongly Disagree", ..., "Strongly Agree")`).
`title`	Optional string. Main chart title.
`subtitle`	Optional string. Chart subtitle.
`x_label`	Optional string. X-axis label. Defaults to empty in crosstab mode or "Variable" in multi-variable mode.
`y_label`	Optional string. Y-axis label. Defaults to "Count" or "Percentage".
`stack_label`	Optional string. Title for the stack legend. Set to NULL, NA, FALSE, or "" to hide the legend title.

stacked_type	One of "normal", "counts" (both show raw counts), or "percent" (100% stacked). Defaults to "counts".
tooltip	A tooltip configuration created with tooltip(), OR a format string with {placeholders}. Available placeholders: {category}, {value}, {series}, {percent}.
tooltip_prefix	Optional string prepended to tooltip values.
tooltip_suffix	Optional string appended to tooltip values.
x_tooltip_suffix	Optional string appended to x-axis values in tooltips.
color_palette	Optional character vector of colors for the stacks.
stack_order	Optional character vector specifying order of stack levels.
x_order	Optional character vector specifying order of x-axis levels.
include_na	Logical. If TRUE, NA values are shown as explicit categories. If FALSE (default), rows with NA are excluded.
na_label_x	String. Label for NA values on x-axis. Default "(Missing)".
na_label_stack	String. Label for NA values in stacks. Default "(Missing)".
x_breaks	Optional numeric vector of cut points for binning x_var.
x_bin_labels	Optional character vector of labels for x_breaks bins.
x_map_values	Optional named list to remap x-axis values for display.
stack_breaks	Optional numeric vector of cut points for binning stack variable.
stack_bin_labels	Optional character vector of labels for stack_breaks bins.
stack_map_values	Optional named list to remap stack values for display.
show_var_tooltip	Logical. If TRUE, shows custom tooltip with variable labels.
horizontal	Logical. If TRUE, creates horizontal bars. Default FALSE.
weight_var	Optional string. Name of a weight variable for weighted counts.
data_labels_enabled	Logical. If TRUE, show value labels on bars. Default TRUE.

Value

A highcharter stacked bar chart object.

See Also

viz_stackedbar for the unified function

Examples

```
# The old way (still works):
# viz_stackedbars(data, x_vars = c("q1", "q2", "q3"))

# The new preferred way:
# viz_stackedbar(data, x_vars = c("q1", "q2", "q3"))
```

viz_timeline *Create a Timeline Chart*

Description

Creates interactive timeline visualizations showing changes in survey responses over time, or simple line charts for pre-aggregated time series data. Supports multiple chart types including stacked areas, line charts, and diverging bar charts.

Usage

```
viz_timeline(
  data,
  time_var,
  y_var,
  group_var = NULL,
  agg = c("percentage", "mean", "sum", "none"),
  chart_type = "line",
  title = NULL,
  subtitle = NULL,
  x_label = NULL,
  y_label = NULL,
  y_max = NULL,
  y_min = NULL,
  color_palette = NULL,
  y_levels = NULL,
  y_breaks = NULL,
  y_bin_labels = NULL,
  y_map_values = NULL,
  y_filter = NULL,
  y_filter_combine = TRUE,
  y_filter_label = NULL,
  time_breaks = NULL,
  time_bin_labels = NULL,
  weight_var = NULL,
  include_na = FALSE,
  na_label_y = "(Missing)",
  na_label_group = "(Missing)",
  tooltip = NULL,
  tooltip_prefix = "",
  tooltip_suffix = "",
  group_order = NULL,
  cross_tab_filter_vars = NULL,
  title_map = NULL,
  legend_position = NULL,
  data_labels_enabled = FALSE,
  backend = "highcharter"
```

```
)
```

Arguments

data	A data frame containing time series data.
time_var	Character string. Name of the time variable (e.g., "year", "wave").
y_var	Character string. Name of the response/value variable.
group_var	Optional character string. Name of grouping variable for separate series (e.g., "country", "gender"). Creates separate lines/areas for each group.
agg	Character string specifying aggregation method: • "percentage" (default): Count responses and calculate percentages per time period. Use for survey data with categorical responses. • "mean": Calculate mean of y_var per time period (and group if specified). • "sum": Calculate sum of y_var per time period (and group if specified). • "none": Use values directly without aggregation. Use for pre-aggregated data where each row represents one observation per time/group combination.
chart_type	Character string. Type of chart: "line" (default) or "stacked_area".
title	Optional main title for the chart.
subtitle	Optional subtitle for the chart.
x_label	Optional character string. Label for the x-axis. Defaults to time_var name.
y_label	Optional character string. Label for the y-axis. Defaults to "Percentage" for agg = "percentage", or y_var name for other modes.
y_max	Optional numeric value. Maximum value for the Y-axis.
y_min	Optional numeric value. Minimum value for the Y-axis.
color_palette	Optional character vector of color hex codes for the series.
y_levels	Optional character vector specifying order of response categories.
y_breaks	Optional numeric vector for binning numeric response values (e.g., c(0, 2.5, 5, 7) to create bins 0-2.5, 2.5-5, 5-7).
y_bin_labels	Optional character vector of labels for response bins (e.g., c("Low (1-2)", "Medium (3-5)", "High (6-7)")).
y_map_values	Optional named list to rename response values for display (e.g., list("1" = "Correct", "0" = "Incorrect")). Applied to legend labels and data.
y_filter	Optional numeric or character vector specifying which response values to include. For numeric responses, use a range like 5:7 to show only values 5, 6, and 7. For categorical responses, use category names like c("Agree", "Strongly Agree"). Applied BEFORE binning (filters raw values first, then bins the filtered data).
y_filter_combine	Logical. When y_filter is used, should filtered values be combined into a single percentage? Defaults to TRUE (show combined % of all filtered values). Set to FALSE to show separate lines for each filtered value.

y_filter_label	Character string. Custom label for the filtered responses in the legend. Only used when y_filter and y_filter_combine = TRUE. If NULL (default) and group_var is provided, shows only group names in legend (e.g., "AgeGroup1"). If NULL and no group_var, uses auto-generated label (e.g., "5-7" for y_filter = 5:7).
time_breaks	Optional numeric vector for binning continuous time variables.
time_bin_labels	Optional character vector of labels for time bins.
weight_var	Optional string. Name of a weight variable for weighted calculations.
include_na	Logical. If TRUE, NA values are included as explicit categories. Default FALSE.
na_label_y	Character string. Label for NA values in the response variable. Default "(Missing)".
na_label_group	Character string. Label for NA values in the group variable. Default "(Missing)".
tooltip	A tooltip configuration created with tooltip(), OR a format string with {placeholders}. Available placeholders: {x}, {y}, {value}, {series}, {percent}. See tooltip for full customization options.
tooltip_prefix	Optional string prepended to values in tooltip.
tooltip_suffix	Optional string appended to values in tooltip.
group_order	Optional character vector specifying display order of group levels.
cross_tab_filter_vars	Character vector. Variables for cross-tab filtering (typically auto-detected from sidebar inputs).
title_map	Named list mapping variable names to custom display titles for dynamic title updates when filtering by cross-tab variables.
legend_position	Position of the legend ("top", "bottom", "left", "right", "none")
data_labels_enabled	Logical. If TRUE, display data point values directly on the chart. Default FALSE.
backend	Rendering backend: "highcharter" (default), "plotly", "echarts4r", or "ggiraph".

Value

A highcharter plot object.

Examples

```
## Not run:
# Load GSS data
data(gss_all)

# Basic timeline - confidence in institutions over time
plot1 <- viz_timeline(
  data = gss_all,
  time_var = "year",
```

```
  y_var = "confinan",
  title = "Confidence in Financial Institutions Over Time",
  y_max = 100
)
plot1

# Line chart by gender
plot2 <- viz_timeline(
  data = gss_all,
  time_var = "year",
  y_var = "happy",
  group_var = "sex",
  chart_type = "line",
  title = "Happiness Trends by Gender",
  y_levels = c("very happy", "pretty happy", "not too happy")
)
plot2

# Show only high responses (5-7 on a 1-7 scale) - COMBINED
plot3 <- viz_timeline(
  data = survey_data,
  time_var = "wave",
  y_var = "agreement",
  group_var = "age_group",
  chart_type = "line",
  y_filter = 5:7,
  title = "% High Agreement (5-7) Over Time"
)
plot3

## End(Not run)
```

`viz_treemap` *Create a treemap visualization*

Description

Creates an interactive treemap using highcharter for hierarchical data. Treemaps are useful for showing proportional data in a space-efficient way.

Usage

```
viz_treemap(
  data,
  group_var,
  subgroup_var = NULL,
  value_var,
  color_var = NULL,
  title = NULL,
```

```
  subtitle = NULL,
  color_palette = NULL,
  height = 500,
  allow_drill_down = TRUE,
  layout_algorithm = "squarified",
  data_labels_enabled = TRUE,
  show_labels = NULL,
  label_style = NULL,
  tooltip = NULL,
  tooltip_format = NULL,
  credits = FALSE,
  pre_aggregated = FALSE,
  legend_position = NULL,
  backend = "highcharter",
  ...
)
```

Arguments

`data`	Data frame containing the data
`group_var`	Primary grouping variable (e.g., "region")
`subgroup_var`	Optional secondary grouping variable (e.g., "city")
`value_var`	Variable for sizing the rectangles (e.g., "spend")
`color_var`	Optional variable for coloring (defaults to group_var)
`title`	Chart title
`subtitle`	Chart subtitle
`color_palette`	Named vector of colors or palette name
`height`	Chart height in pixels (default 500)
`allow_drill_down`	Whether to allow drilling into subgroups (default TRUE)
`layout_algorithm`	Layout algorithm: "squarified" (default), "strip", "sliceAndDice", "stripes"
`data_labels_enabled`	Logical. If TRUE, show data labels on cells. Default TRUE.
`show_labels`	Deprecated. Use `data_labels_enabled` instead.
`label_style`	List of label styling options
`tooltip`	A tooltip configuration created with `tooltip()`, OR a format string with {placeholders}. Available placeholders: `{name}`, `{value}`. See `tooltip` for full customization options.
`tooltip_format`	Custom tooltip format using Highcharts syntax (legacy). For the simpler dashboardr placeholder syntax, use `tooltip` instead.
`credits`	Whether to show Highcharts credits (default FALSE)
`pre_aggregated`	Logical. If TRUE, skips summing and uses `value_var` directly. Use this when your data is already aggregated (one row per leaf node). Default is FALSE.

legend_position	Position of the legend ("top", "bottom", "left", "right", "none")
backend	Rendering backend: "highcharter" (default), "plotly", "echarts4r", or "ggiraph".
...	Additional parameters passed to highcharter

Value

A highcharter treemap object

Examples

```
## Not run:
# Simple treemap
data <- data.frame(
  region = c("North", "North", "South", "South"),
  city = c("NYC", "Boston", "Miami", "Atlanta"),
  spend = c(1000, 500, 800, 600)
)
viz_treemap(data, group_var = "region", subgroup_var = "city", value_var = "spend")

# Single-level treemap
viz_treemap(data, group_var = "city", value_var = "spend", title = "Spend by City")

## End(Not run)
```

---

viz_waffle *Create a Waffle Chart*

Description

Creates an interactive waffle (square pie) chart using highcharter. Waffle charts display proportional data as a grid of colored squares, providing an intuitive alternative to pie charts.

Usage

```
viz_waffle(
  data,
  x_var,
  y_var = NULL,
  title = NULL,
  subtitle = NULL,
  total = 100,
  rows = 10,
  color_palette = NULL,
  x_order = NULL,
  data_labels_enabled = FALSE,
  show_in_legend = TRUE,
  weight_var = NULL,
```

```
  border_color = "white",
  border_width = 1,
  tooltip = NULL,
  tooltip_prefix = "",
  tooltip_suffix = "",
  height = 400,
  backend = "highcharter"
)
```

Arguments

data	A data frame containing the data.
x_var	Character string. Name of the categorical variable (category labels).
y_var	Optional character string. Name of a numeric column with pre-aggregated values (counts or percentages). If NULL, counts are computed from the data.
title	Optional main title for the chart.
subtitle	Optional subtitle for the chart.
total	Numeric. Total number of squares in the waffle grid. Default 100. Each square represents `total_value / total` of the data.
rows	Numeric. Number of rows in the waffle grid. Default 10.
color_palette	Optional character vector of colors for the categories.
x_order	Optional character vector specifying the order of categories.
data_labels_enabled	Logical. If TRUE, show category labels. Default FALSE (legend is shown instead).
show_in_legend	Logical. If TRUE (default), show a legend.
weight_var	Optional character string. Name of weight variable.
border_color	Character string. Color of the square borders. Default "white".
border_width	Numeric. Width of square borders in pixels. Default 1.
tooltip	A tooltip configuration created with `tooltip()`.
tooltip_prefix	Optional string prepended to tooltip values.
tooltip_suffix	Optional string appended to tooltip values.
height	Numeric. Chart height in pixels. Default 400.
backend	Rendering backend: "highcharter" (default), "plotly", "echarts4r", or "ggiraph".

Value

A highcharter plot object.

Examples

```
## Not run:
df <- data.frame(
  category = c("Agree", "Neutral", "Disagree"),
  count = c(45, 30, 25)
)
viz_waffle(df, x_var = "category", y_var = "count",
           title = "Survey Responses", total = 100)

## End(Not run)
```

Index