Package 'rtemis.draw'

November 3, 2026

Title Schema-Driven Interactive Charts, Networks, Maps, and Diagrams

Version 0.5.4

Date 2026-11-02

Description Create interactive statistical charts, network graphs, choropleth maps, and diagrams. Charts are rendered with the 'ECharts' 'JavaScript' library, networks with 'Sigma.js', maps with 'MapLibre GL JS', and diagrams written in 'Mermaid' syntax with 'Pele'. A high-level interface of plotting functions builds a complete visualization from data in a single call, while a low-level interface gives full control over every option. Both are based on the same type-checked and validated 'S7' configuration objects. Configurations can be saved to and restored from JSON files, and each chart type has a published JSON Schema that other interfaces can implement. Visualizations display as 'htmlwidgets' that automatically adapt to light and dark themes, work in R Markdown, 'Quarto', and 'shiny', and can be exported to Scalable Vector Graphics (SVG) files using 'Node.js'.

URL https://www.rtemis.org, https://docs.rtemis.org/r/draw, https://docs.rtemis.org/r/draw-api

BugReports https://github.com/rtemis-org/draw/issues

License BSD_3_clause + file LICENSE

Encoding UTF-8

Language en-US

SystemRequirements Node.js (>= 18) for optional SVG export

Depends R (>= 4.4.0)

Imports htmltools, htmlwidgets, jsonlite, rtemis.core (>= 0.4.7), S7, signal, stats, tools, utils, viridisLite

Suggests data.table, jsonvalidate (>= 1.5.0), mgcv, pROC, rmarkdown, rtemis, rtemis.a3 (>= 0.6.0), shiny, survival, systemfonts, testthat (>= 3.0.0)

Config/testthat/edition 3

Config/testthat/parallel true

Collate 'rtemis.draw-package.R' 'rtemis_color_system.R' 'utils.R' 'styles.R' 'label.R' 'axis.R' 'components.R' 'series.R' 'series_sankey.R' 'option.R' 'theme.R' 'legend_layout.R' 'zoom.R' 'draw.R' 'chart_config.R' 'config_constraints.R' 'config_scatter.R' 'config_bar.R' 'config_density.R' 'config_histogram.R' 'distribution.R' 'annotations.R' 'fit_layer.R' 'fit_model.R' 'scatter_input.R' 'config_line.R' 'timeseries.R' 'scatter3d.R' 'surface_layer.R' 'config_pie.R' 'config_boxplot.R' 'config_sankey.R' 'config_gantt.R' 'chart_io.R' 'schema.R' 'network.R' 'map.R' 'config_network.R' 'config_choropleth.R' 'mermaid.R' 'config_mermaid.R' 'heatmap_input.R' 'config_heatmap.R' 'draw_a3.R' 'draw_protein.R' 'config_a3.R' 'draw_spectrogram.R' 'config_spectrogram.R' 'draw_gantt.R' 'draw_fit.R' 'draw_learning_curve.R' 'draw_varimp.R' 'config_significance.R' 'draw_significance.R' 'config_confusion.R' 'draw_confusion.R' 'config_roc.R' 'draw_roc.R' 'config_survival.R' 'draw_survival.R' 'draw_survfit.R' 'config_calibration.R' 'draw_calibration.R' 'boxplot_layers.R' 'draw_boxplot.R' 'draw_metric.R' 'draw_panels.R' 'draw_harmonograph.R' 'export.R' 'zzz.R'

Config/roxygen2/version 8.1.1

NeedsCompilation no

Author E.D. Gennatas [aut, cre, cph] (ORCID: <https://orcid.org/0000-0001-9280-3609>)

Maintainer E.D. Gennatas <gennatas@gmail.com>

Repository CRAN

Date/Publication 2026-11-03 04:00:02 UTC

Contents

rtemis.draw-package 6
A3Config 8
AnnotationLayer 10
AreaStyle 11
Axis 12
AxisLabel 14
AxisLine 15
AxisTick 16
BarConfig 17
BarSeries 19
BoxplotConfig 20
BoxplotSeries 23
CalibrationConfig 25
ChartConfig 27

chart_config_to_list . . . 28
chart_dispatcher_schema . . . 29
chart_registry . . . 30
chart_schema . . . 31
ChoroplethConfig . . . 32
compile . . . 33
ConfusionConfig . . . 34
confusion_input . . . 36
DataZoom . . . 37
DensityConfig . . . 40
draw . . . 42
drawOutput . . . 44
draw_a3 . . . 45
draw_add_fit . . . 47
draw_add_surface . . . 48
draw_annotate . . . 49
draw_bar . . . 50
draw_boxplot . . . 52
draw_calibration . . . 56
draw_choropleth . . . 57
draw_confusion . . . 60
draw_density . . . 61
draw_fit . . . 63
draw_gantt . . . 66
draw_graph . . . 68
draw_harmonograph . . . 70
draw_heatmap . . . 72
draw_histogram . . . 76
draw_learning_curve . . . 79
draw_line . . . 81
draw_manhattan . . . 84
draw_map . . . 86
draw_mermaid . . . 88
draw_metric . . . 91
draw_network . . . 93
draw_panels . . . 96
draw_pie . . . 97
draw_protein . . . 98
draw_pvals . . . 99
draw_roc . . . 100
draw_sankey . . . 102
draw_scatter . . . 104
draw_scatter3d . . . 107
draw_spectrogram . . . 108
draw_survfit . . . 111
draw_survival . . . 112
draw_ts . . . 114
draw_varimp . . . 115

draw_violin . . . 118
draw_volcano . . . 118
draw_xt . . . 120
EChartsOption . . . 122
FitLayer . . . 124
GanttConfig . . . 125
GraphEdge . . . 127
GraphModel . . . 128
GraphNode . . . 128
Grid . . . 129
HeatmapConfig . . . 130
HeatmapSeries . . . 133
HistogramConfig . . . 134
ItemStyle . . . 137
LabelLine . . . 139
LabelOption . . . 140
Legend . . . 141
LineConfig . . . 143
LineSeries . . . 145
LineStyle . . . 147
MapLibreOption . . . 148
MapModel . . . 150
MapRow . . . 151
MarkArea . . . 152
MarkAreaDataPoint . . . 153
MermaidConfig . . . 154
MinorSplitLine . . . 155
MinorTick . . . 156
NetworkConfig . . . 157
PanelLayout . . . 159
panel_layout_schema . . . 159
PeleOption . . . 160
PieConfig . . . 161
PieLabelOption . . . 163
PieSeries . . . 164
read_chart_config . . . 166
renderDraw . . . 167
resolve . . . 168
ROCConfig . . . 169
roc_probabilities . . . 171
roc_vertices . . . 172
SankeyConfig . . . 172
SankeyEdgeItem . . . 174
SankeyLevelOption . . . 175
SankeyNodeItem . . . 176
SankeySeries . . . 177
save_drawing . . . 179
Scatter3DConfig . . . 180

ScatterConfig . . . 182
ScatterSeries . . . 185
setup_A3Config . . . 186
setup_BarConfig . . . 188
setup_BoxplotConfig . . . 190
setup_CalibrationConfig . . . 193
setup_ChoroplethConfig . . . 195
setup_ConfusionConfig . . . 196
setup_DensityConfig . . . 198
setup_GanttConfig . . . 200
setup_HeatmapConfig . . . 202
setup_HistogramConfig . . . 205
setup_LineConfig . . . 208
setup_MermaidConfig . . . 210
setup_NetworkConfig . . . 212
setup_PanelLayout . . . 214
setup_PieConfig . . . 214
setup_ROCConfig . . . 216
setup_SankeyConfig . . . 218
setup_Scatter3DConfig . . . 219
setup_ScatterConfig . . . 221
setup_SignificanceConfig . . . 224
setup_SpectrogramConfig . . . 227
setup_SurvivalConfig . . . 229
setup_TimeSeriesConfig . . . 231
SigmaOption . . . 233
SignificanceConfig . . . 235
SpectrogramConfig . . . 238
SplitArea . . . 240
SplitLine . . . 241
summarize_varimp . . . 242
SurfaceLayer . . . 243
SurvivalConfig . . . 244
TextStyle . . . 246
Theme . . . 249
theme_dark . . . 250
theme_light . . . 252
TimeSeriesConfig . . . 254
Title . . . 256
Tooltip . . . 258
to_json . . . 260
to_list . . . 260
true_pred_data . . . 261
varimp_data . . . 261
VisualMap . . . 262
write_chart_config . . . 263

Index **265**

`rtemis.draw-package` *rtemis.draw: Schema-Driven Interactive Charts, Networks, Maps, and Diagrams*

Description

Create interactive statistical charts, network graphs, choropleth maps, and diagrams. Charts are rendered with 'ECharts', networks with 'Sigma.js', maps with 'MapLibre GL JS', and diagrams written in 'Mermaid' syntax with 'Pele'. Every visualization is an 'htmlwidgets' widget that automatically follows the light or dark theme of wherever it is shown, and can be exported to a Scalable Vector Graphics (SVG) file.

Statistical charts

- Distributions and comparisons: `draw_scatter()`, `draw_line()`, `draw_bar()`, `draw_boxplot()`, `draw_violin()`, `draw_histogram()`, `draw_density()`, `draw_pie()`, `draw_heatmap()`, `draw_sankey()`.
- Time: `draw_ts()`, `draw_xt()`, `draw_gantt()`, `draw_spectrogram()`.
- Three dimensions: `draw_scatter3d()`, with `draw_add_surface()`.
- Model evaluation: `draw_roc()`, `draw_calibration()`, `draw_confusion()`, `draw_fit()`, `draw_varimp()`, `draw_metric()`, `draw_learning_curve()`.
- Time-to-event: `draw_survival()`, `draw_survfit()`.
- Significance testing: `draw_volcano()`, `draw_manhattan()`, `draw_pvals()`.
- Sequences: `draw_protein()`, `draw_a3()`.

Add fitted lines, reference lines, bands, and labels with `draw_add_fit()` and `draw_annotate()`, and arrange several charts into one figure with `draw_panels()`.

Network graphs

`draw_network()` draws a network from an adjacency or correlation matrix, or from an edge list, with 'Sigma.js'.

Choropleth maps

`draw_choropleth()` colors countries, U.S. states, or U.S. counties by a value, with 'MapLibre GL JS'. Boundaries are bundled with the package, so maps need no network access.

Diagrams

`draw_mermaid()` draws a diagram from 'Mermaid' text with 'Pele', including flowcharts, sequence, class, state, and entity relationship diagrams, Gantt charts, mindmaps, and every other diagram type built into 'Mermaid' 12.1.0. `setup_MermaidConfig()` also reads the diagram text from a `.mmd` file.

Configuration

Each `draw_*()` function builds a validated 'S7' configuration object, e.g. from `setup_ScatterConfig()`, and renders it. Configurations can be written to and read from JSON files with `write_chart_config()` and `read_chart_config()`, and `chart_schema()` returns the JSON Schema that such files validate against. `draw()` renders a complete low-level option object directly.

Themes

By default, visualizations detect whether they are shown on a light or dark background in 'RStudio', 'VS Code', 'Quarto', and the browser, style themselves to match, and switch when the viewer changes theme, e.g. with the 'Quarto' light/dark toggle or the operating system setting. Networks and maps keep their current pan and zoom when they switch. Pass `theme_light()` or `theme_dark()` as `theme` to fix a theme.

Options

- `rtemis.font_family`: Font family of every visualization, e.g. `options(rtemis.font_family = "Geist")`, followed by the system UI font as the fallback. It is the default `font_family` of `theme_light()` and `theme_dark()`, so it applies to the plots of every rtemis package drawn with 'rtemis.draw'. SVG export of diagrams measures labels in this font when it is installed and the 'systemfonts' package is available.

Output

Visualizations display in the 'RStudio' and 'VS Code' viewers, R Markdown and 'Quarto' documents, and 'shiny' apps (see `drawOutput()` and `renderDraw()`). All 'JavaScript' is bundled with the package, so HTML output works offline. `save_drawing()` exports a visualization to an SVG file using 'Node.js'.

Author(s)

Maintainer: E.D. Gennatas <gennatas@gmail.com> (ORCID) [copyright holder]

Authors:

- E.D. Gennatas <gennatas@gmail.com> (ORCID) [copyright holder]

See Also

Useful links:

- https://www.rtemis.org
- https://docs.rtemis.org/r/draw
- https://docs.rtemis.org/r/draw-api
- Report bugs at https://github.com/rtemis-org/draw/issues

A3Config *Annotated Protein Diagram Configuration*

Description

A serializable description of an A3 protein diagram. Build one with `setup_A3Config()` rather than calling this constructor directly.

Usage

```
A3Config(
  dat_path = NULL,
  title = NULL,
  origin = NULL,
  writer = NULL,
  n_per_row = 21L,
  position_every = 10L,
  ptm_placement = "radial",
  zoom = TRUE,
  residue_spacing = 0.3,
  marker_size = 28,
  font_size = 18,
  line_width = 2,
  show_markers = TRUE,
  show_labels = TRUE,
  region_opacity = 0.3,
  residue_fill = NULL,
  residue_stroke = NULL,
  label_color = NULL,
  pos_label_color = NULL,
  variant_color = NULL,
  disease_variant_color = NULL
)
```

Arguments

`dat_path`	Optional Character: Path to the data, read at draw time. The serializable alternative to passing `data` to `draw()`.
`title`	Optional Character: Chart title. Declared here because it is the one property every chart type has; `palette` is not (four take a `colormap` instead), and axis labels and margins are cartesian-only.
`origin`	Optional Named character {"user", "default", "derived"}: Where each value came from, one entry per settable property. Absent on an authored config; written by the interface that resolved it.
`writer`	Optional Named character: Which interface wrote the config, as `name` and `version`. Absent on an authored config.

`n_per_row` Integer `[1, Inf)`: Residues per row.

`position_every` Integer `[1, Inf)`: Label every nth position.

`ptm_placement` Character: How post-translational modifications are placed relative to the backbone.

`zoom` Logical: Enable Shift+wheel zoom, and wheel and drag panning, along the sequence.

`residue_spacing` Numeric `[0, Inf)`: Gap between residues.

`marker_size` Numeric `[0, Inf)`: Preferred residue marker size; reduced with related marks when needed to fit the surface.

`font_size` Numeric `[0, Inf)`: Residue label font size.

`line_width` Numeric `[0, Inf)`: Backbone line width.

`show_markers`, `show_labels` Logical: Whether to draw each.

`region_opacity` Numeric `[0, 1]`: Region band opacity.

`residue_fill`, `residue_stroke` Optional Character: Residue marker colors.

`label_color`, `pos_label_color` Optional Character: Label colors.

`variant_color`, `disease_variant_color` Optional Character: Variant marker colors. `NULL` selects the corresponding light/dark A3 theme color.

Details

The bound data is an `A3` object, created with `rtemis.a3::create_A3()`.

Value

`A3Config` object.

Author(s)

EDG

Examples

```
setup_A3Config(n_per_row = 30L)@type
```

`AnnotationLayer` *Cartesian Annotation Layer*

Description

Portable, materialized annotation geometry. Corresponds to MarkLineOption, MarkAreaOption, and ScatterSeriesOption in ECharts src/component/marker and src/chart/scatter. Coordinates are data values (epoch milliseconds for time axes); categorical coordinates may be category names. Layers are independent of group legend selection.

Usage

```
AnnotationLayer(
  hline = NULL,
  vline = NULL,
  x = numeric(0),
  y = numeric(0),
  text = NULL,
  bands = list(),
  color = "#888888",
  line_type = "dashed",
  opacity = 0.15,
  x_axis = 0L,
  y_axis = 0L
)
```

Arguments

`hline`, `vline`	Numeric: Horizontal and vertical reference positions.
`x`, `y`	Numeric or Character: Coordinates for text annotations.
`text`	Character: One label per coordinate pair.
`bands`	List: Numeric length-two x intervals to shade.
`color`	Character: Annotation color.
`line_type`	Character `{"solid", "dashed", "dotted"}`: Reference line style.
`opacity`	Numeric `[0, 1]`: Band opacity.
`x_axis`, `y_axis`	Integer `[0, Inf)`: Zero-based target axis indices.

Value

AnnotationLayer: Validated annotation geometry.

Examples

```
AnnotationLayer(hline = 0, text = "Peak", x = 2, y = 3)
```

AreaStyle *Area Style*

Description

Fill styling for areas (polygons, area under curves, etc.).

Usage

```
AreaStyle(
  color = NULL,
  opacity = NULL,
  origin = NULL,
  shadow_blur = NULL,
  shadow_color = NULL,
  shadow_offset_x = NULL,
  shadow_offset_y = NULL
)
```

Arguments

color	Optional Character: CSS fill color.
opacity	Optional Numeric [0, 1]: Fill opacity.
origin	Optional Character {"auto", "start", "end"} or Numeric: Area origin.
shadow_blur	Optional Numeric [0, Inf): Shadow blur radius.
shadow_color	Optional Character: Shadow color.
shadow_offset_x	Optional Numeric: Shadow horizontal offset.
shadow_offset_y	Optional Numeric: Shadow vertical offset.

Details

Corresponds to AreaStyleOption in src/util/types.ts (line 1169). ECharts docs: https://echarts.apache.org/en/option.html#series-line.areaStyle

Value

AreaStyle object.

Examples

```
AreaStyle(color = "#16a085", opacity = 0.3)
```

Axis *Axis*

Description

Configuration for a cartesian axis (xAxis or yAxis). Combines fields from `AxisBaseOptionCommon`, `NumericAxisBaseOptionCommon`, `CategoryAxisBaseOption`, `ValueAxisBaseOption`, `LogAxisBaseOption`, and `TimeAxisBaseOption`.

Usage

```
Axis(
  type = NULL,
  show = NULL,
  inverse = NULL,
  name = NULL,
  name_location = NULL,
  name_rotate = NULL,
  name_gap = NULL,
  name_text_style = NULL,
  silent = NULL,
  trigger_event = NULL,
  min = NULL,
  max = NULL,
  scale = NULL,
  split_number = NULL,
  interval = NULL,
  min_interval = NULL,
  max_interval = NULL,
  align_ticks = NULL,
  log_base = NULL,
  boundary_gap = NULL,
  data = NULL,
  axis_line = NULL,
  axis_tick = NULL,
  minor_tick = NULL,
  axis_label = NULL,
  split_line = NULL,
  minor_split_line = NULL,
  split_area = NULL,
  axis_pointer = NULL,
  position = NULL,
  grid_index = NULL
)
```

Arguments

`type`	Optional Character {"value", "category", "time", "log"}: Axis type.

show	Optional Logical: Whether to show the axis.
inverse	Optional Logical: Whether to invert the axis.
name	Optional Character: Axis name.
name_location	Optional Character {"start", "middle", "center", "end"}: Axis name location.
name_rotate	Optional Numeric: Axis name rotation angle in degrees.
name_gap	Optional Numeric `[0, Inf)`: Gap between the axis name and axis line.
name_text_style	Optional TextStyle: Axis name text style.
silent	Optional Logical: Whether the axis is silent.
trigger_event	Optional Logical: Whether the axis triggers events.
min	Optional Numeric or Character {"dataMin", "dataMax"}: Minimum axis value.
max	Optional Numeric or Character {"dataMin", "dataMax"}: Maximum axis value.
scale	Optional Logical: Whether a value axis should avoid forcing zero.
split_number	Optional Numeric `[0, Inf)`: Suggested number of split segments.
interval	Optional Numeric `[0, Inf)`: Tick interval.
min_interval	Optional Numeric `[0, Inf)`: Minimum auto-calculated tick interval.
max_interval	Optional Numeric `[0, Inf)`: Maximum auto-calculated tick interval.
align_ticks	Optional Logical: Whether to align ticks with the first axis.
log_base	Optional Numeric `[0, Inf)`: Logarithm base for log axes.
boundary_gap	Optional Logical, Numeric, or Character: Gap at axis boundaries. Logical (category) or length-2 vector of numbers/strings (numeric axes).
data	Optional Vector: Category data.
axis_line	Optional AxisLine: Axis line configuration.
axis_tick	Optional AxisTick: Axis tick configuration.
minor_tick	Optional MinorTick: Minor tick configuration.
axis_label	Optional AxisLabel: Axis label configuration.
split_line	Optional SplitLine: Split line configuration.
minor_split_line	Optional MinorSplitLine: Minor split line configuration.
split_area	Optional SplitArea: Split area configuration.
axis_pointer	Optional list: Interactive mouse-following axis guide (`axisPointer`), e.g. `list(show = TRUE, type = "line")`. Uses echarts-cased keys.
position	Optional Character {"left", "right", "top", "bottom"}: Axis position. For `yAxis`: `"left"` (default) or `"right"`. For `xAxis`: `"bottom"` (default) or `"top"`.
grid_index	Optional Numeric `[0, Inf)`: Index of the grid this axis belongs to. Used when the chart has multiple grids (e.g. heatmap + dendrogram panels).

Details

Corresponds to `AxisBaseOption` in `src/coord/axisCommonTypes.ts` (line 376). ECharts docs: https://echarts.apache.org/en/option.html#xAxis

Value

`Axis` object.

Examples

```
Axis(type = "category", data = c("Mon", "Tue", "Wed"))
```

`AxisLabel` *Axis Label*

Description

Configuration for axis labels. Combines label-specific positioning with text styling (flattened in to_list like LabelOption).

Usage

```
AxisLabel(
  show = NULL,
  inside = NULL,
  rotate = NULL,
  margin = NULL,
  formatter = NULL,
  show_min_label = NULL,
  show_max_label = NULL,
  hide_overlap = NULL,
  interval = NULL,
  text_style = NULL
)
```

Arguments

`show`	Optional Logical: Whether to show axis labels.
`inside`	Optional Logical: Whether labels are inside the grid.
`rotate`	Optional Numeric: Label rotation angle in degrees.
`margin`	Optional Numeric `[0, Inf)`: Space between the label and axis line in pixels.
`formatter`	Optional Character or function: Label formatter.
`show_min_label`	Optional Logical: Whether to show the label at the minimum end.
`show_max_label`	Optional Logical: Whether to show the label at the maximum end.
`hide_overlap`	Optional Logical: Whether to hide overlapping labels.
`interval`	Optional Numeric or Character {"auto"}: Label interval.
`text_style`	Optional TextStyle: Text appearance.

Details

Corresponds to `AxisLabelBaseOption` in `src/coord/axisCommonTypes.ts` (line 312). ECharts docs: https://echarts.apache.org/en/option.html#xAxis.axisLabel

Value

`AxisLabel` object.

Examples

```
AxisLabel(show = TRUE, rotate = 45, margin = 8)
```

AxisLine *Axis Line*

Description

Configuration for the axis line itself.

Usage

```
AxisLine(
  show = NULL,
  on_zero = NULL,
  on_zero_axis_index = NULL,
  symbol = NULL,
  symbol_size = NULL,
  symbol_offset = NULL,
  line_style = NULL
)
```

Arguments

`show`	Optional Logical or Character {"auto"}: Whether to show the axis line.
`on_zero`	Optional Logical: Whether the axis line is on the zero position of the other axis.
`on_zero_axis_index`	Optional Numeric `[0, Inf)`: Index of the other axis on whose zero position this axis line is drawn.
`symbol`	Optional Character: Arrow symbols at both ends.
`symbol_size`	Optional Numeric: Arrow symbol size.
`symbol_offset`	Optional Numeric or Character: Arrow symbol offset.
`line_style`	Optional LineStyle: Line styling.

Details

Corresponds to `AxisLineOption` in `src/coord/axisCommonTypes.ts` (line 231). ECharts docs: https://echarts.apache.org/en/option.html#xAxis.axisLine

Value

AxisLine object.

Examples

```
AxisLine(show = TRUE, line_style = LineStyle(color = "#888888"))
```

AxisTick *Axis Tick*

Description

Configuration for axis tick marks.

Usage

```
AxisTick(
  show = NULL,
  inside = NULL,
  length = NULL,
  align_with_label = NULL,
  line_style = NULL
)
```

Arguments

show	Optional Logical or Character {"auto"}: Whether to show ticks.
inside	Optional Logical: Whether ticks are inside the grid.
length	Optional Numeric [0, Inf): Tick mark length in pixels.
align_with_label	Optional Logical: Whether to align ticks with labels.
line_style	Optional LineStyle: Line styling.

Details

Corresponds to AxisTickOption in src/coord/axisCommonTypes.ts (line 244). ECharts docs: https://echarts.apache.org/en/option.html#xAxis.axisTick

Value

AxisTick object.

Examples

```
AxisTick(show = TRUE, length = 5, align_with_label = TRUE)
```

BarConfig *Bar Chart Configuration*

Description

A serializable description of a bar chart. Build one with `setup_BarConfig()` rather than calling this constructor directly.

Usage

```
BarConfig(
  dat_path = NULL,
  title = NULL,
  origin = NULL,
  writer = NULL,
  legend_position = "top",
  legend_placement = "outside",
  legend_title = NULL,
  x = NULL,
  y = NULL,
  order = "input",
  stack = FALSE,
  horizontal = FALSE,
  zoom = FALSE,
  bar_width = NULL,
  palette = NULL,
  xlab = NULL,
  ylab = NULL,
  margin_top = NULL,
  margin_right = NULL,
  margin_bottom = NULL,
  margin_left = NULL
)
```

Arguments

`dat_path`	Optional Character: Path to the data, read at draw time. The serializable alternative to passing `data` to `draw()`.
`title`	Optional Character: Chart title. Declared here because it is the one property every chart type has; `palette` is not (four take a `colormap` instead), and axis labels and margins are cartesian-only.
`origin`	Optional Named character {"user", "default", "derived"}: Where each value came from, one entry per settable property. Absent on an authored config; written by the interface that resolved it.
`writer`	Optional Named character: Which interface wrote the config, as `name` and `version`. Absent on an authored config.

`legend_position`
: Character {"top", "bottom", "left", "right", "top-left", "top-right", "bottom-left", "bottom-right"}: Legend anchor. Top/bottom anchors use horizontal rows; left/right anchors use a vertical column. Corner anchors align within the top or bottom row.

`legend_placement`
: Character {"outside", "inside"}: Relation to the plotting area. Outside placement reserves space for the complete legend; inside placement overlays the data. Neither setting adds a missing legend.

`legend_title`
: Optional Character: Title drawn with the legend or colorbar: above left/right legends and vertical colorbars, before the first row of top/bottom legends and horizontal colorbars. `NULL` draws no title. A title does not add a missing legend.

`x`
: Optional Character: Column holding the categories.

`y`
: Optional Character: Columns holding the values, one series each.

`order`
: Character `{"input", "increasing", "decreasing"}`: Order categories by their total across series, retaining ties and placing missing totals last.

`stack`
: Logical: If TRUE, stack the series instead of grouping them.

`horizontal`
: Logical: If TRUE, draw the bars horizontally.

`zoom`
: Logical: Enable a slider, Shift+wheel zoom, and drag panning on the category axis.

`bar_width`
: Optional Numeric `(0, Inf)`: Bar thickness in pixels. NULL lets ECharts size bars from the category spacing.

`palette`
: Optional Character: Series colors, overriding the theme palette for this chart. `NULL` uses the theme's.

`xlab`, `ylab`
: Optional Character: Axis labels. `NULL` derives them from the data.

`margin_top`, `margin_right`, `margin_bottom`, `margin_left`
: Optional Integer `[0, Inf)`: Plot margins in pixels.

Details

`x` names the column holding the categories. `y` names **one or more** columns, one series each – a single column draws one set of bars, several draw a grouped or stacked chart with the column names as the series names. That is the difference from ScatterConfig, whose `x` and `y` each name exactly one column.

Fixed thickness maps to `barWidth` in ECharts' `BaseBarSeriesOption` (`src/chart/bar/BaseBarSeries.ts`). It applies to either orientation.

Value

`BarConfig` object.

Author(s)

EDG

Examples

```
setup_BarConfig(x = "cyl", y = "mpg")@type
```

`BarSeries` *Bar Series*

Description

Configuration for a bar chart series.

Usage

```
BarSeries(
  name = NULL,
  data = NULL,
  x_axis_index = NULL,
  y_axis_index = NULL,
  stack = NULL,
  silent = NULL,
  legend_hover_link = NULL,
  color = NULL,
  z_level = NULL,
  z = NULL,
  clip = NULL,
  bar_width = NULL,
  bar_max_width = NULL,
  bar_min_width = NULL,
  bar_min_height = NULL,
  bar_gap = NULL,
  bar_category_gap = NULL,
  round_cap = NULL,
  show_background = NULL,
  item_style = NULL,
  label = NULL
)
```

Arguments

`name`	Optional Character: Series name for legend display.
`data`	Optional Vector or list: Data values.
`x_axis_index`	Optional Numeric `[0, Inf)`: X-axis index.
`y_axis_index`	Optional Numeric `[0, Inf)`: Y-axis index.
`stack`	Optional Character: Stack group name.
`silent`	Optional Logical: Whether the series is silent.

`legend_hover_link`
: Optional Logical: Whether to highlight related legends on hover.

`color`
: Optional Character: Series color override.

`z_level`
: Optional Numeric: Canvas layer index.

`z`
: Optional Numeric: Front-back order within the same layer.

`clip`
: Optional Logical: Whether to clip overflow.

`bar_width`
: Optional Numeric or Character: Bar width.

`bar_max_width`
: Optional Numeric or Character: Maximum bar width.

`bar_min_width`
: Optional Numeric or Character: Minimum bar width.

`bar_min_height`
: Optional Numeric `[0, Inf)`: Minimum bar height.

`bar_gap`
: Optional Numeric or Character: Gap between bars in the same category.

`bar_category_gap`
: Optional Numeric or Character: Gap between categories.

`round_cap`
: Optional Logical: Whether to use a round cap.

`show_background`
: Optional Logical: Whether to show the bar background.

`item_style`
: Optional ItemStyle: Bar styling.

`label`
: Optional LabelOption: Data-label configuration.

Details

Corresponds to `BarSeriesOption` in `src/chart/bar/BarSeries.ts` (line 68) and `BaseBarSeriesOption` in `src/chart/bar/BaseBarSeries.ts` (line 38). ECharts docs: https://echarts.apache.org/en/option.html#series-bar

Value

`BarSeries` object.

Examples

```
BarSeries(name = "Counts", data = c(3, 7, 2))
```

BoxplotConfig *Boxplot Configuration*

Description

A serializable description of a boxplot. Build one with `setup_BoxplotConfig()` rather than calling this constructor directly.

Usage

```
BoxplotConfig(
  dat_path = NULL,
  title = NULL,
  origin = NULL,
  writer = NULL,
  legend_position = "top",
  legend_placement = "outside",
  legend_title = NULL,
  x = NULL,
  group = NULL,
  observation = NULL,
  order = "input",
  transform = "none",
  geometry = "box",
  bandwidth = NULL,
  adjust = 1,
  density_points = 128L,
  paired = FALSE,
  pair_alpha = 0.35,
  pair_width = 1,
  comparisons = NULL,
  quartiles = "linear",
  whisker = 1.5,
  boxpoints = "none",
  point_size = 5,
  point_alpha = 0.6,
  point_spread = 0.5,
  horizontal = FALSE,
  zoom = FALSE,
  na_rm = TRUE,
  labels = NULL,
  palette = NULL,
  fill_alpha = 0.25,
  xlab = NULL,
  ylab = NULL,
  margin_top = NULL,
  margin_right = NULL,
  margin_bottom = NULL,
  margin_left = NULL
)
```

Arguments

dat_path	Optional Character: Path to the data, read at draw time. The serializable alternative to passing data to draw().
title	Optional Character: Chart title.

`origin` Optional Named character {"user", "default", "derived"}: Where each value came from, one entry per settable property. Absent on an authored config; written by the interface that resolved it.

`writer` Optional Named character: Which interface wrote the config, as `name` and `version`. Absent on an authored config.

`legend_position`
Character {"top", "bottom", "left", "right", "top-left", "top-right", "bottom-left", "bottom-right"}: Legend anchor. Top/bottom anchors use horizontal rows; left/right anchors use a vertical column. Corner anchors align within the top or bottom row.

`legend_placement`
Character {"outside", "inside"}: Relation to the plotting area. Outside placement reserves space for the complete legend; inside placement overlays the data. Neither setting adds a missing legend.

`legend_title` Optional Character: Title drawn with the legend or colorbar: above left/right legends and vertical colorbars, before the first row of top/bottom legends and horizontal colorbars. `NULL` draws no title. A title does not add a missing legend.

`x` Optional Character: Columns to summarize, one box each.

`group` Optional Character: Column that splits each box by level.

`observation` Optional Character: Column identifying observations for tooltips and pairing.

`order` Character `{"input", "mean", "median"}`: Keep input order or sort categories by decreasing mean/median after transformation; empty boxes last.

`transform` Character `{"none", "scale", "minmax"}`: Transform each input column before grouping. Scale subtracts the mean and divides by the sample standard deviation; minmax maps the available range to zero through one. Constant columns become zero and missing values remain missing.

`geometry` Character {"box", "violin", "both"}: Distribution geometry.

`bandwidth` Optional Numeric `(0, Inf)`: Finite Gaussian kernel bandwidth.

`adjust` Numeric `(0, Inf)`: Finite bandwidth multiplier.

`density_points` Integer `[16, 4096]`: Density evaluation grid size.

`paired` Logical: Connect explicit IDs across adjacent categories.

`pair_alpha` Numeric `[0, 1]`: Paired line opacity.

`pair_width` Numeric `(0, Inf)`: Finite paired line width in pixels.

`comparisons` Optional Character: Name of a comparison data frame in the bound data list. Its columns follow `draw_boxplot()`.

`quartiles` Character {"linear", "hinges"}: Quartile convention.

`whisker` Numeric `[0, Inf)`: Finite IQR fence multiplier; zero uses full-range whiskers.

`boxpoints` Character {"none", "all", "outliers"}: Values to overlay.

`point_size` Numeric `(0, Inf)`: Finite point diameter in pixels.

`point_alpha` Numeric `[0, 1]`: Point opacity.

`point_spread` Numeric `[0, 1]`: Fraction of box width occupied by offsets.

horizontal	Logical: Draw the boxes horizontally.
zoom	Logical: Enable a slider, Shift+wheel zoom, and drag panning on the category axis.
na_rm	Logical: Remove missing values. FALSE rejects missing input.
labels	Optional Character: Box labels. NULL uses the bound column names.
palette	Optional Character: Box colors, overriding the theme palette for this chart. NULL uses the package's box colors.
fill_alpha	Numeric [0, 1]: Box fill opacity.
xlab, ylab	Optional Character: Axis labels.
margin_top, margin_right, margin_bottom, margin_left	Optional Integer [0, Inf): Plot margins in pixels.

Details

x names **one or more** columns, one box each. group optionally names a column that splits every box into one per level. Statistics and point placement follow draw_boxplot(). Compiles to BoxplotSeriesOption and CustomSeriesOption in ECharts src/chart/boxplot/BoxplotSeries.ts and src/chart/custom/CustomSeries.ts.

Value

BoxplotConfig object.

Author(s)

EDG

Examples

```
setup_BoxplotConfig(x = c("mpg", "hp"))@type
```

BoxplotSeries *Boxplot Series*

Description

Configuration for a boxplot chart series.

Usage

```
BoxplotSeries(
  name = NULL,
  data = NULL,
  x_axis_index = NULL,
  y_axis_index = NULL,
  silent = NULL,
```

```
  legend_hover_link = NULL,
  color = NULL,
  z_level = NULL,
  z = NULL,
  layout = NULL,
  box_width = NULL,
  item_style = NULL,
  label = NULL
)
```

Arguments

name	Optional Character: Series name.
data	Optional list: Boxplot data. Each element is c(min, Q1, median, Q3, max).
x_axis_index	Optional Numeric [0, Inf): X-axis index.
y_axis_index	Optional Numeric [0, Inf): Y-axis index.
silent	Optional Logical: Whether the series is silent.
legend_hover_link	Optional Logical: Whether to highlight related legends on hover.
color	Optional Character: Series color override.
z_level	Optional Numeric: Canvas layer index.
z	Optional Numeric: Front-back order within the same layer.
layout	Optional Character {"horizontal", "vertical"}: Layout orientation.
box_width	Optional Numeric or Character: Box-width range as a length-2 vector.
item_style	Optional ItemStyle: Boxplot styling.
label	Optional LabelOption: Data-label configuration.

Details

Corresponds to BoxplotSeriesOption in src/chart/boxplot/BoxplotSeries.ts (line 59). ECharts docs: https://echarts.apache.org/en/option.html#series-boxplot

Value

BoxplotSeries object.

Examples

```
# Each element is one box, given as its five-number summary.
BoxplotSeries(
  name = "Distribution",
  data = list(c(850, 940, 960, 980, 1070), c(750, 800, 840, 890, 1020))
)
```

`CalibrationConfig` *Calibration Plot Configuration*

Description

Bind binary observation records to a reliability diagram. Each row contains an observed outcome (zero or one) and its predicted probability, optionally grouped by sample or model. Configuration carries column names, not data.

Usage

```
CalibrationConfig(
  dat_path = NULL,
  title = NULL,
  origin = NULL,
  writer = NULL,
  legend_position = "top",
  legend_placement = "outside",
  legend_title = NULL,
  observed = "observed",
  probability = "probability",
  group = NULL,
  n_bins = 10L,
  bin_method = "quantile",
  na_rm = TRUE,
  mode = "lines+markers",
  show_brier = TRUE,
  rug = TRUE,
  rug_size = 8,
  rug_opacity = 0.3,
  point_size = 7,
  line_width = 2,
  digits = 3L,
  diagonal = TRUE,
  diagonal_color = "#888888",
  palette = NULL,
  legend = TRUE,
  square = TRUE,
  xlab = "Mean predicted probability",
  ylab = "Observed proportion"
)
```

Arguments

`dat_path`	Optional Character: Path to the data, read at draw time. The serializable alternative to passing `data` to `draw()`.
`title`	Optional Character: Chart title.

`origin` Optional Named character {"user", "default", "derived"}: Where each value came from, one entry per settable property. Absent on an authored config; written by the interface that resolved it.

`writer` Optional Named character: Which interface wrote the config, as `name` and `version`. Absent on an authored config.

`legend_position` Character {"top", "bottom", "left", "right", "top-left", "top-right", "bottom-left", "bottom-right"}: Legend anchor. Top/bottom anchors use horizontal rows; left/right anchors use a vertical column. Corner anchors align within the top or bottom row.

`legend_placement` Character {"outside", "inside"}: Relation to the plotting area. Outside placement reserves space for the complete legend; inside placement overlays the data. Neither setting adds a missing legend.

`legend_title` Optional Character: Title drawn with the legend or colorbar: above left/right legends and vertical colorbars, before the first row of top/bottom legends and horizontal colorbars. `NULL` draws no title. A title does not add a missing legend.

`observed` Character: Column containing binary outcomes, zero or one.

`probability` Character: Column containing predicted probabilities.

`group` Optional Character: Column identifying samples or models.

`n_bins` Integer `[1, Inf)`: Requested number of bins per group.

`bin_method` Character {"quantile", "equidistant"}: Binning rule.

`na_rm` Logical: Remove incomplete observation/probability pairs.

`mode` Character {"lines", "markers", "lines+markers"}: Curve display.

`show_brier` Logical: Include each sample's Brier score in its legend label.

`rug` Logical: Show the distribution of individual probabilities.

`rug_size` Numeric `(0, Inf)`: Rug tick height in pixels.

`rug_opacity` Numeric `[0, 1]`: Rug tick opacity.

`point_size` Numeric `(0, Inf)`: Calibration marker diameter in pixels.

`line_width` Numeric: Curve stroke width in pixels.

`digits` Integer: Decimal places for AUC labels and tooltip values.

`diagonal` Logical: Show an independent chance diagonal.

`diagonal_color` Character: Chance-line color.

`palette` Optional Character: Group colors; unset uses the chart theme.

`legend` Logical: Show group labels and AUC summaries.

`square` Logical: Keep the plotting grid square.

`xlab` Character: Horizontal axis label.

`ylab` Character: Vertical axis label.

Details

Compiles to `LineSeriesOption` in `src/chart/line/LineSeries.ts` and `ScatterSeriesOption` in `src/chart/scatter/ScatterSeries.ts`. ECharts docs: https://echarts.apache.org/en/option.html#series-line

Value

A `CalibrationConfig` object.

Statistical semantics

Bins are computed independently within each group. Equidistant bins partition the unit interval. Quantile bins use linear interpolation at positions `(n - 1) * p + 1` in the sorted probabilities (R quantile type 7). Repeated boundaries are collapsed without splitting tied scores. Intervals include their lower boundary and exclude their upper boundary, except that the final interval includes its upper boundary. Constant probabilities form one bin. Empty bins are omitted; lines join the remaining bin means. Each point is the mean predicted probability and mean observed outcome in its bin. The Brier score is the mean squared probability error across all complete observations, before binning. It is not an average of bin errors. Missing pairs are removed together when `na_rm` is true; invalid finite ranges and infinite values are always rejected. A group with no complete observations is rejected. The probability rug uses those same complete observations and sits just inside the lower plotting edge. Both axes span zero to one. Display precision does not round plotted values. A one-bin curve remains visible as a point even in lines-only mode.

Examples

```
config <- setup_CalibrationConfig(n_bins = 2L, bin_method = "equidistant")
draw(config, data = data.frame(observed = c(0, 1), probability = c(.2, .8)))
```

ChartConfig *Chart Configuration Base Class*

Description

Abstract base for every chart config. Subclasses override `type` with a constant, which is the discriminator the published schema dispatches on.

Usage

```
ChartConfig(
  type = NULL,
  dat_path = NULL,
  title = NULL,
  origin = NULL,
  writer = NULL
)
```

Arguments

`type`	Character: Chart type. Set by each subclass, not by the user.
`dat_path`	Optional Character: Path to the data, read at draw time. The serializable alternative to passing `data` to `draw()`.

`title`	Optional Character: Chart title. Declared here because it is the one property every chart type has; `palette` is not (four take a `colormap` instead), and axis labels and margins are cartesian-only.
`origin`	Optional Named character {"user", "default", "derived"}: Where each value came from, one entry per settable property. Absent on an authored config; written by the interface that resolved it.
`writer`	Optional Named character: Which interface wrote the config, as `name` and `version`. Absent on an authored config.

Value

`ChartConfig` object.

Author(s)

EDG

Examples

```
# ChartConfig is abstract: build one of its subclasses.
config <- setup_ScatterConfig(x = "wt", y = "mpg")
S7::S7_inherits(config, ChartConfig)
```

---

chart_config_to_list *Convert a chart config to a plain list*

Description

Convert a chart config to a plain list

Usage

```
chart_config_to_list(config, complete = FALSE)
```

Arguments

`config`	ChartConfig: The chart configuration.
`complete`	Logical: If TRUE, emit the output-config form: every property, plus provenance.

Details

Two forms, matching the two the schema publishes:

- `complete = FALSE` (default) drops unset properties, so an authored config stays as small as it was authored. `type` is always kept: it carries the document's shape and is the one key the schema requires.
- `complete = TRUE` keeps every property, unset ones included, so they serialize as explicit nulls – and stamps the `writer`. A consumer of such a document can read every key directly instead of reproducing this package's defaults. Requires a full `origin` map; see `write_chart_config()`.

Value

Named list, ready for `jsonlite::toJSON()`.

Author(s)

EDG

Examples

```
chart_config_to_list(setup_ScatterConfig(x = "wt", y = "mpg"))
```

`chart_dispatcher_schema` *Generate the chart family dispatcher schema*

Description

The family's entry point: a document is a chart if it matches exactly one of the leaf schemas, selected by its `type`.

Usage

```
chart_dispatcher_schema(classes, id, leaf_ids, title, description)
```

Arguments

`classes`	List: ChartConfig subclasses.
`id`	Character: The dispatcher's `$id` URL.
`leaf_ids`	Character: `$id` of each leaf schema, in the same order as `classes`.
`title`	Character: Human-readable schema title.
`description`	Character: What the family is.

Value

Named list: The JSON Schema, ready for `jsonlite::toJSON()`.

Author(s)

EDG

Examples

```
schema <- chart_dispatcher_schema(
  classes = list(ScatterConfig, BarConfig),
  id = "https://schemas.rtemis.org/draw/chart.schema.json",
  leaf_ids = c(
    "https://schemas.rtemis.org/draw/scatter.schema.json",
    "https://schemas.rtemis.org/draw/bar.schema.json"
  ),
  title = "Chart",
  description = "Any rtemis.draw chart."
)
schema[["properties"]][["type"]][["enum"]]
```

---

chart_registry *Chart type registry*

Description

Maps each chart `type` to the class that models it and the `setup_*` that builds one. Read at run time by `read_chart_config()`, and at generation time by `data-raw/generate_schemas.R`, so the two cannot disagree about which chart types exist.

Usage

```
chart_registry()
```

Value

Named list, one entry per chart type, each with `cls` and `setup`.

Author(s)

EDG

Examples

```
names(chart_registry())
```

chart_schema *Generate a JSON Schema for a chart config class*

Description

Emits the schema for one ChartConfig subclass: every property it declares or inherits, typed from its `PropertySpec`, plus a constant for the `type` discriminator. The result is self-contained – it validates a chart document on its own, without the family dispatcher.

Usage

```
chart_schema(cls, id, title, description, complete = FALSE)
```

Arguments

`cls`	S7 class: A ChartConfig subclass.
`id`	Character: The schema's `$id` URL.
`title`	Character: Human-readable schema title.
`description`	Character: What this chart is.
`complete`	Logical: If TRUE, emit the output-config kind, requiring every property.

Details

Two kinds, differing only in what they require:

- `complete = FALSE` (an **input config**) requires only `type`. The author sets a subset and the interface fills in the rest.
- `complete = TRUE` (an **output config**) requires every property, so a document that claims to be complete can be checked rather than trusted. The chart's own properties may still be null – "this chart has no title" is a fact worth recording – but the provenance maps may not: a document that cannot say where its values came from is not complete, whatever it claims.

Neither emits a `default`: what an interface fills in is not a fact about the document. Shared structural class rules are emitted in `allOf`, including inherited rules. Input schemas allow omitted values to be resolved by an interface; record schemas require every value. Numeric ordering, bound-column contents, and statistical/data-dependent checks still run during construction or compilation and cannot be inferred from JSON Schema validation alone.

Value

Named list: The JSON Schema, ready for `jsonlite::toJSON()`.

Author(s)

EDG

Examples

```
schema <- chart_schema(
  ScatterConfig,
  id = "https://schema.rtemis.org/chart/scatter/v1/schema.json",
  title = "rtemis ScatterConfig",
  description = "Scatter chart."
)
names(schema[["properties"]])
```

ChoroplethConfig *Choropleth Map Configuration*

Description

A serializable description of a choropleth map. Build one with `setup_ChoroplethConfig()` rather than calling this constructor directly.

Usage

```
ChoroplethConfig(
  dat_path = NULL,
  title = NULL,
  origin = NULL,
  writer = NULL,
  location = NULL,
  value = NULL,
  tooltip = NULL,
  resolution = "country",
  classification = "quantile",
  num_classes = 5,
  colormap = "blues",
  opacity = 1,
  show_boundaries = TRUE,
  outline_width = 0.2,
  show_legend = TRUE,
  legend_position = "bottom-right",
  tooltip_position = "top-right",
  value_label = NULL
)
```

Arguments

`dat_path`	Optional Character: Path to the data, read at draw time. The serializable alternative to passing `data` to `draw()`.
`title`	Optional Character: Chart title. Declared here because it is the one property every chart type has; `palette` is not (four take a `colormap` instead), and axis labels and margins are cartesian-only.

`origin`	Optional Named character {"user", "default", "derived"}: Where each value came from, one entry per settable property. Absent on an authored config; written by the interface that resolved it.
`writer`	Optional Named character: Which interface wrote the config, as `name` and `version`. Absent on an authored config.
`location`	Optional Character: Column holding the region identifiers.
`value`	Optional Character: Column holding the value to shade by.
`tooltip`	Optional Character: Column holding per-region tooltip text.
`resolution`	Character {"country", "state", "county"}: Which boundary set the locations refer to.
`classification`	Character {"quantile", "equal", "jenks"}: How values are binned into classes.
`num_classes`	Numeric `[2, 12]`: Number of color classes.
`colormap`	Character: Continuous color scheme for the classes.
`opacity`	Numeric `[0, 1]`: Fill opacity.
`show_boundaries`, `show_legend`	Logical: Whether to draw each.
`outline_width`	Numeric `[0, Inf)`: Boundary line width.
`legend_position`, `tooltip_position`	Character: Corner each overlay is anchored to.
`value_label`	Optional Character: Label for the value in the legend and tooltip. `NULL` derives it from the bound column.

Value

`ChoroplethConfig` object.

Author(s)

EDG

Examples

```
setup_ChoroplethConfig(location = "st", value = "v")@type
```

`compile` *Compile a chart config into a render option*

Description

Turns a ChartConfig into the backend option object `draw()` renders – an EChartsOption for ECharts charts. This is where a config's column names are resolved against actual data.

Usage

```
compile(config, data = NULL, theme = NULL, ...)
```

Arguments

config	ChartConfig: The chart configuration.
data	Optional Data frame or named list: The data to plot. When NULL, the config's dat_path is read instead.
theme	Optional Theme, list, or NA: The theme the option will be drawn with, as in draw(). Charts that compute colors before the widget exists read them from it: sign colors a config leaves unset, and the background a color scale fades to. NULL uses the built-in light and dark themes.
...	Passed to methods.

Details

The generic does the two steps every chart needs before its own translation can start, so that no method can forget one: it materializes the data (from data, or from the config's dat_path) and resolve()s the config against it. Methods therefore always receive a **resolved** config and **non-NULL** data, and are pure translation.

Value

Render option object, e.g. EChartsOption.

Author(s)

EDG

Examples

```
config <- setup_ScatterConfig(x = "wt", y = "mpg")
option <- compile(config, data = mtcars)
option@x_axis@name
```

ConfusionConfig *Confusion Plot Configuration*

Description

Portable confusion counts with reference rows and predicted columns. Data consists of long records with class labels and a frequency, optionally grouped into panels. Missing class labels represent omitted pairs; absent class combinations have zero frequency. All panels use the same class order. Compiles to HeatmapSeriesOption in src/chart/heatmap/HeatmapSeries.ts, GridOption in src/coord/cartesian/GridModel.ts, and continuous visual maps. to_list() emits semantic config keys; compile() emits ECharts options. draw() fits square count cells with aligned marginal summaries to the available canvas. Missing-pair omissions are reported in the console. The cell counts show included observations without a separate sample-size label. Color fades and marginal backgrounds follow the active theme unless explicitly overridden.

Usage

```
ConfusionConfig(
  dat_path = NULL,
  title = NULL,
  origin = NULL,
  writer = NULL,
  reference = "reference",
  predicted = "predicted",
  count = "n",
  panel = NULL,
  classes = NULL,
  show_metrics = TRUE,
  digits = 2L,
  ncol = 2L,
  correct_color = "#0F6A66",
  incorrect_color = "#BE2E5F",
  low_color = NULL,
  summary_color = NULL,
  font_size = 12,
  xlab = "Predicted",
  ylab = "Reference"
)
```

Arguments

dat_path	Optional Character: Path to the data, read at draw time. The serializable alternative to passing data to draw().
title	Optional Character: Chart title. Declared here because it is the one property every chart type has; palette is not (four take a colormap instead), and axis labels and margins are cartesian-only.
origin	Optional Named character {"user", "default", "derived"}: Where each value came from, one entry per settable property. Absent on an authored config; written by the interface that resolved it.
writer	Optional Named character: Which interface wrote the config, as name and version. Absent on an authored config.
reference	Character: Column containing reference class labels.
predicted	Character: Column containing predicted class labels.
count	Character: Column containing nonnegative integer frequencies.
panel	Optional Character: Optional panel-label column; unset draws one matrix.
classes	Optional Character vector: Ordered class labels shared by every panel; unset preserves first appearance.
show_metrics	Logical: Show per-class rates, accuracy, and balanced accuracy.
digits	Integer: Decimal places for rates in cells and hover text.
ncol	Integer: Maximum number of panels per row.
correct_color	Character: Six-digit hex color at unit fraction for correct predictions.

incorrect_color
: Character: Six-digit hex color at unit fraction for incorrect predictions.

low_color
: Optional Character: Six-digit hex color at zero row fraction; unset uses the active theme background.

summary_color
: Optional Character: Six-digit hex background for metric cells; unset uses a faint neutral tint of the active theme background.

font_size
: Numeric: Preferred cell-label font size in pixels. Constrained surfaces reduce text to fit, down to 8 pixels. Long class labels wrap and column labels rotate when needed. Increase figure dimensions if labels and metrics cannot fit without overlap.

xlab
: Character: Predicted-class axis label.

ylab
: Character: Reference-class axis label.

Value

A `ConfusionConfig` object.

Statistical semantics

Color intensity is the fraction within each reference row; labels show raw counts. Sensitivity, specificity, PPV, and NPV are one-versus-rest rates for each named class. Balanced accuracy is mean class recall, including every declared class. Zero denominators produce NA, never zero; consequently balanced accuracy is NA if any declared class has no reference observations. Summaries of pooled resamples describe pooled predictions, not mean fold performance. Repeated observations are counted each time they appear.

Examples

```
cfg <- setup_ConfusionConfig()
draw(cfg, data = data.frame(reference = c("yes", "no"),
  predicted = c("yes", "no"), n = c(8, 12)))
```

---

confusion_input *Normalize confusion inputs to portable frequency records*

Description

Retains missing-label frequencies so omission is visible after serialization. Matrix dimnames carry class identity; columns are matched by name, never by an assumed diagonal position. No model dependency is needed for data inputs.

Usage

```
confusion_input(x, y = NULL, classes = NULL)
```

Arguments

x	Matrix, table, data frame, list of matrices, or vector: Counts or labels.
y	Optional vector: Predicted labels paired with reference labels in x.
classes	Optional Character vector: Explicit class order.

Value

Data frame with reference, predicted, n, and optionally panel columns.

Examples

```
confusion_input(factor(c("yes", "no")), factor(c("yes", "yes")))
```

DataZoom *Data Zoom*

Description

Axis-range zoom component. ECharts renders dataZoom as an array; common usage pairs a "slider" (visible scrollbar) with an "inside" (mouse-wheel / drag) entry targeting the same axis. Unused styling / mouse-behavior fields stay NULL and are dropped from the serialized output.

Usage

```
DataZoom(
  type = "slider",
  id = NULL,
  disabled = NULL,
  x_axis_index = NULL,
  y_axis_index = NULL,
  radius_axis_index = NULL,
  angle_axis_index = NULL,
  single_axis_index = NULL,
  filter_mode = NULL,
  start = NULL,
  end = NULL,
  start_value = NULL,
  end_value = NULL,
  min_span = NULL,
  max_span = NULL,
  min_value_span = NULL,
  max_value_span = NULL,
  orient = NULL,
  throttle = NULL,
  range_mode = NULL,
  show = NULL,
```

```
  background_color = NULL,
  left = NULL,
  right = NULL,
  top = NULL,
  bottom = NULL,
  width = NULL,
  height = NULL,
  zoom_lock = NULL,
  zoom_on_mouse_wheel = NULL,
  move_on_mouse_move = NULL,
  move_on_mouse_wheel = NULL,
  prevent_default_mouse_move = NULL
)
```

Arguments

type	Character {"slider", "inside"}: Zoom type. "slider" shows a draggable handle; "inside" enables zoom/drag directly on the coordinate system.
id	Optional Character: Component identifier used for updates.
disabled	Optional Logical: Whether the component is disabled.
x_axis_index	Optional Numeric, numeric vector, or Character {"all"}: Index of the x-axis (or axes) this zoom targets.
y_axis_index	Optional Numeric, numeric vector, or Character {"all"}: Index of the y-axis (or axes) this zoom targets.
radius_axis_index	Optional Numeric, numeric vector, or Character {"all"}: Polar radius axis index.
angle_axis_index	Optional Numeric, numeric vector, or Character {"all"}: Polar angle axis index.
single_axis_index	Optional Numeric, numeric vector, or Character {"all"}: Single-axis index.
filter_mode	Optional Character {"filter", "weakFilter", "empty", "none"}: How out-of-window data points are handled.
start	Optional Numeric [0, 100]: Left/start position as a percentage of the full data range.
end	Optional Numeric [0, 100]: Right/end position as a percentage of the full data range.
start_value	Optional Numeric, Character, or Date: Absolute start value (alternative to start).
end_value	Optional Numeric, Character, or Date: Absolute end value (alternative to end).
min_span	Optional Numeric [0, 100]: Minimum window size, as a percentage.
max_span	Optional Numeric [0, 100]: Maximum window size, as a percentage.
min_value_span	Optional Numeric or Character: Minimum window size in data units.
max_value_span	Optional Numeric or Character: Maximum window size in data units.
orient	Optional Character {"horizontal", "vertical"}: Layout orientation. Defaults are inferred from the axis the zoom targets.

throttle Optional Numeric [0, Inf): Throttle delay in milliseconds for zoom events.

range_mode Optional length-2 Character vector of {"value", "percent"}: Per-boundary interpretation for start/end.

show Optional Logical: Whether the slider is visible (slider type only).

background_color Optional Character: Slider background color.

left Optional Numeric or Character: Slider left position.

right Optional Numeric or Character: Slider right position.

top Optional Numeric or Character: Slider top position.

bottom Optional Numeric or Character: Slider bottom position.

width Optional Numeric or Character: Slider width.

height Optional Numeric or Character: Slider height.

zoom_lock Optional Logical: Whether to lock the window size (allow move but not zoom).

zoom_on_mouse_wheel Optional Logical or Character {"shift", "ctrl", "alt"}: Whether/how the mouse wheel zooms (inside type only).

move_on_mouse_move Optional Logical or Character {"shift", "ctrl", "alt"}: Whether/how mouse drag pans (inside type only).

move_on_mouse_wheel Optional Logical or Character {"shift", "ctrl", "alt"}: Whether/how the mouse wheel pans (inside type only).

prevent_default_mouse_move Optional Logical: Whether to call `event.preventDefault()` on drag (inside type only).

Details

Corresponds to `DataZoomOption` in `src/component/dataZoom/DataZoomModel.ts` (line 38), `SliderDataZoomOption` in `src/component/dataZoom/SliderZoomModel.ts` (line 38), and `InsideDataZoomOption` in `src/component/dataZoom/InsideZoomModel.ts` (line 23). ECharts docs: https://echarts.apache.org/en/option.html#dataZoom

Value

`DataZoom` object.

Examples

```
DataZoom(type = "slider", start = 0, end = 50)
```

DensityConfig *Density Chart Configuration*

Description

A serializable description of a kernel density chart. Build one with `setup_DensityConfig()` rather than calling this constructor directly.

Usage

```
DensityConfig(
  dat_path = NULL,
  title = NULL,
  origin = NULL,
  writer = NULL,
  legend_position = "top",
  legend_placement = "outside",
  legend_title = NULL,
  x = NULL,
  group = NULL,
  mode = "overlap",
  order = "input",
  zoom = FALSE,
  n = 512L,
  bw = "nrd0",
  bandwidth = NULL,
  kernel = "gaussian",
  adjust = 1,
  fill_alpha = 0.25,
  na_rm = TRUE,
  palette = NULL,
  xlab = NULL,
  ylab = NULL,
  margin_top = NULL,
  margin_right = NULL,
  margin_bottom = NULL,
  margin_left = NULL
)
```

Arguments

`dat_path`	Optional Character: Path to the data, read at draw time. The serializable alternative to passing `data` to `draw()`.
`title`	Optional Character: Chart title. Declared here because it is the one property every chart type has; `palette` is not (four take a `colormap` instead), and axis labels and margins are cartesian-only.

`origin` Optional Named character {"user", "default", "derived"}: Where each value came from, one entry per settable property. Absent on an authored config; written by the interface that resolved it.

`writer` Optional Named character: Which interface wrote the config, as `name` and `version`. Absent on an authored config.

`legend_position`
Character {"top", "bottom", "left", "right", "top-left", "top-right", "bottom-left", "bottom-right"}: Legend anchor. Top/bottom anchors use horizontal rows; left/right anchors use a vertical column. Corner anchors align within the top or bottom row.

`legend_placement`
Character {"outside", "inside"}: Relation to the plotting area. Outside placement reserves space for the complete legend; inside placement overlays the data. Neither setting adds a missing legend.

`legend_title` Optional Character: Title drawn with the legend or colorbar: above left/right legends and vertical colorbars, before the first row of top/bottom legends and horizontal colorbars. `NULL` draws no title. A title does not add a missing legend.

`x` Optional Character: Columns to estimate, one distribution per column.

`group` Optional Character: Column to split the estimate by.

`mode` Character `{"overlap", "ridge"}`: Overlay samples or align rows on common x and y scales. Ridge rows use sample names in place of a legend. Increase figure height for many rows.

`order` Character `{"input", "mean", "median"}`: Sample order. Summary orders are decreasing, retain ties in input order, and place empty samples last.

`zoom` Logical: Enable a slider, Shift+wheel zoom, and drag panning on the x-axis. Ridge rows zoom together, without a slider.

`n` Integer `[2, Inf)`: Points at which the density is estimated.

`bw` Character or Numeric: Bandwidth selector (`"nrd0"`, `"nrd"`, `"ucv"`, `"bcv"`, `"SJ"`, `"SJ-ste"`, `"SJ-dpi"`). Numeric input is normalized to `bandwidth` by the setup function.

`bandwidth` Optional Numeric `(0, Inf)`: Explicit finite smoothing bandwidth in measurement units, overriding the selector.

`kernel` Character: One of `"gaussian"`, `"epanechnikov"`, `"rectangular"`, `"triangular"`, `"biweight"`, `"cosine"`, or `"optcosine"`.

`adjust` Numeric `(0, Inf)`: Finite bandwidth multiplier.

`fill_alpha` Numeric `[0, 1]`: Distribution fill opacity.

`na_rm` Logical: If TRUE, drop `NA` values before estimating.

`palette` Optional Character: Series colors, overriding the theme palette for this chart. `NULL` uses the theme's.

`xlab`, `ylab` Optional Character: Axis labels. `NULL` derives them from the data.

`margin_top`, `margin_right`, `margin_bottom`, `margin_left`
Optional Integer `[0, Inf)`: Plot margins in pixels.

Details

x names one or more numeric columns. group optionally names a column to split it by, drawing one curve per level. The output uses LineSeriesOption in ECharts src/chart/line/LineSeries.ts.

Value

DensityConfig object.

Author(s)

EDG

Examples

```
setup_DensityConfig(x = "mpg")@type
```

---

draw *Render a chart configuration or backend option as an htmlwidget*

Description

draw() renders chart configurations and backend option objects as interactive visualizations. It is an 'S7' generic that dispatches on the class of option:

Usage

```
draw(
  option,
  theme = NULL,
  width = NULL,
  height = NULL,
  element_id = NULL,
  filename = NULL,
  animation = NULL,
  ...
)
```

Arguments

option	ChartConfig subclass, EChartsOption, SigmaOption, MapLibreOption, PeleOption, or named list: Chart configuration or render spec to draw. All concrete chart configuration classes listed below are supported.
theme	Optional Theme, list, or NA: Theme override. NULL enables light/dark auto-detection; NA disables theming. Applied to every backend.
width	Optional Character or Numeric: Widget width.
height	Optional Character or Numeric: Widget height.

`element_id` Optional Character: Explicit element ID.

`filename` Optional Character: If provided, the widget is also written to this file via `save_drawing()`. Extension determines the format (currently only `.svg`).

`animation` Optional Logical: Whether ECharts animates the chart. `NULL` (default) leaves ECharts' own default in place. `FALSE` disables animation, which matters for charts with many points, where animating every interactive update is expensive. Ignored by the Sigma, MapLibre, and Pele backends, which have no ECharts animation to disable.

`...` Named arguments for the selected method. For ChartConfig subclasses, `data` supplies the data to plot; its required structure depends on the configuration class. If `data` is omitted or `NULL`, the config's `dat_path` is read instead.

Additional arguments are passed to the rendering backend. ECharts accepts `renderer` (Character {"canvas", "svg"}) and `meta` (named list of extra payload fields, used internally e.g. by `draw_heatmap()`). Any other name is an error rather than a silent no-op, so a typo cannot quietly change what is drawn.

Two `meta` fields are read by the ECharts JS binding and are part of its contract, because they need the container width, which only the browser knows:

- `squareCells`: see `draw_heatmap()`.
- `aspect`: a named list with `ratio` (required grid height / grid width), `widthPx` (preferred grid width; the box shrinks below this only when the container is too narrow), and `leftPx`, `rightPx`, `topPx`, `botPx` (padding to reserve outside the grid). The binding solves for the grid box on render and on resize, and sets the container height to match. Use it for charts whose *pixel* geometry carries meaning – most often equal axis scaling, where one data unit must be the same number of pixels on both axes and `ratio` is therefore the y range over the x range. ECharts has no equal-aspect option, and R cannot know the container width, so neither side can enforce this alone.

Details

- ChartConfig subclasses -> compile the configuration with its data, then render using the corresponding backend.
- EChartsOption -> 'ECharts' (`rtemis-draw` widget).
- SigmaOption -> 'Sigma.js' (`rtemis-graph` widget).
- MapLibreOption -> 'MapLibre' (`rtemis-map` widget).
- PeleOption -> 'Pele' (`rtemis-diagram` widget).

A plain named list is treated as a raw 'ECharts' option.

Theme handling is uniform across backends and lives on `draw()`, not on the option object: pass `NULL` (default) for light/dark auto-detection (from VS Code, RStudio, or the browser's `prefers-color-scheme`), `NA` for no theme (raw backend defaults), or a Theme / list to force one.

Value

htmlwidget: Widget object.

Chart configurations

Supported ChartConfig subclasses are A3Config, BarConfig, BoxplotConfig, CalibrationConfig, ChoroplethConfig, ConfusionConfig, DensityConfig, GanttConfig, HeatmapConfig, HistogramConfig, LineConfig, MermaidConfig, NetworkConfig, PieConfig, ROCConfig, SankeyConfig, ScatterConfig, Scatter3DConfig, SignificanceConfig, SpectrogramConfig, SurvivalConfig, and TimeSeriesConfig. Use their `setup_*()` functions to create configurations, then pass the data to `draw()` with `data =` or set `dat_path` on the configuration. See `compile()` to obtain the backend option without rendering it.

Examples

```
# Render a chart configuration with data.
config <- setup_ScatterConfig(x = "wt", y = "mpg")
draw(config, data = mtcars)

# Render a backend option directly.
option <- EChartsOption(
  x_axis = Axis(type = "category", data = c("Mon", "Tue", "Wed")),
  y_axis = Axis(type = "value"),
  series = BarSeries(data = c(120, 200, 150))
)
draw(option)
```

---

drawOutput *Shiny output for draw widget*

Description

Shiny output for draw widget

Usage

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

Arguments

`outputId`	Character: Shiny output ID.
`width`	Character or Numeric: CSS width.
`height`	Character or Numeric: CSS height.

Value

Shiny output element for a draw widget.

Examples

```
if (requireNamespace("shiny", quietly = TRUE)) {
  drawOutput("chart", height = "600px")
}
```

draw_a3 *Draw an A3 Amino Acid Sequence Visualization*

Description

Renders an A3 object (from rtemis.a3) as an interactive ECharts amino-acid sequence diagram. The sequence is wrapped into rows in a meander/serpentine path, with optional site, region, PTM, processing, and variant annotations overlaid as distinct series and collected in an upper-right annotation column, matching rtemislive. Legend headings use native rich text and remain available in vector SVG exports through save_drawing().

Usage

```
draw_a3(
  x,
  n_per_row = 21L,
  residue_spacing = 0.3,
  marker_size = 28,
  font_size = 18,
  line_width = 2,
  show_markers = TRUE,
  show_labels = TRUE,
  position_every = 10L,
  region_opacity = 0.3,
  ptm_placement = "radial",
  residue_fill = NULL,
  residue_stroke = NULL,
  label_color = NULL,
  pos_label_color = NULL,
  variant_color = NULL,
  disease_variant_color = NULL,
  zoom = TRUE,
  title = NULL,
  grid = NULL,
  theme = NULL,
  width = NULL,
  height = NULL,
  element_id = NULL,
  filename = NULL,
  enable_zoom = NULL
)
```

Arguments

x	A3 object from rtemis.a3::create_A3() or rtemis.a3::read_A3json().
n_per_row	Integer [2, Inf): Number of residues per wrapped row. Reduce this for larger residues on narrow surfaces.

residue_spacing
: Numeric (0, Inf): Horizontal spacing multiplier. Values below 1 compress the layout; 1 gives natural hex spacing.

marker_size
: Numeric (0, Inf): Preferred residue circle diameter in pixels; reduced with related marks when needed to fit the drawing surface.

font_size
: Numeric (0, Inf): Base font size in pixels for residue and position labels.

line_width
: Numeric (0, Inf): Backbone line width in pixels.

show_markers
: Logical: Whether to render the backbone line and residue circles.

show_labels
: Logical: Whether to render single-letter residue labels.

position_every
: Optional Integer [1, Inf): Show a numeric position label every N residues. NULL disables position labels.

region_opacity
: Numeric [0, 1]: Opacity of region band overlays.

ptm_placement
: Character {"radial", "innerRadial", "outerRadial"}: Placement of PTM symbols relative to the residue circle. "radial" centers on the circle edge; "innerRadial" places inside; "outerRadial" places outside.

residue_fill
: Optional Character: Residue circle fill color.

residue_stroke
: Optional Character: Residue circle stroke and backbone line color.

label_color
: Optional Character: Default residue label text color.

pos_label_color
: Optional Character: Position label text color.

variant_color
: Optional Character: Label color for variant residues.

disease_variant_color
: Optional Character: Label color for disease-associated variant residues (takes precedence over variant_color). NULL color arguments follow the light/dark A3 theme.

zoom
: Logical: Enable Shift+wheel zoom, and wheel and drag panning, along the sequence.

title
: Optional Character: Chart title.

grid
: Optional Grid: Override any plot-area margin. The defaults (margins matching rtemislive, with right driven by legend width) are merged with the properties of the supplied Grid object, so only the fields you set are changed — e.g. Grid(left = 8, top = 8). legend_top is re-derived from the final grid top automatically.

theme
: Optional Theme: Theme override. NULL auto-detects light/dark mode; NA uses raw ECharts defaults.

width
: Optional Numeric or Character: Widget width.

height
: Optional Numeric or Character: Widget height. Auto-computed from layout bounds and marker_size when NULL.

element_id
: Optional Character: Explicit DOM element ID for the widget container. NULL lets htmlwidgets generate one.

filename
: Optional Character: If provided, save the widget to this file via save_drawing().

enable_zoom
: Deprecated Logical: Former name of zoom; a value sets zoom with a warning.

Details

The annotation column stays at the upper right on every surface. Residue markers, labels, and annotations shrink together to avoid overlap. Row wrapping remains fixed. Browser height follows the layout unless supplied; SVG and panel dimensions remain bounded. A custom `grid` retains its layout.

Corresponds to `createA3EChartsOption()` in `src/lib/a3/visualization/echarts.ts`.

Value

htmlwidget: Widget object.

Examples

```
if (requireNamespace("rtemis.a3", quietly = TRUE)) {
  a <- rtemis.a3::create_A3(
    "MAEPRQEFEVMEDHAGTYGLGDRK",
    site = list(
      Active_site = rtemis.a3::annotation_position(c(5L, 17L))
    ),
    region = list(
      Domain = rtemis.a3::annotation_range(
        matrix(c(3L, 10L, 15L, 22L), ncol = 2, byrow = TRUE)
      )
    )
  )
  draw_a3(a)
}
```

---

draw_add_fit *Add a Supplied Fit and Interval to a Drawing*

Description

Accepts predictions from any model without keeping a fitted R object in the chart. Existing axis limits are preserved, including explicitly requested limits. Annotate child drawings before composing panels. For built-in GLM or GAM estimation, use `draw_scatter()` or `draw_fit()`.

Usage

```
draw_add_fit(
  plot,
  x = NULL,
  y = NULL,
  lower = NULL,
  upper = NULL,
  name = "Fit",
  color = "#6ca3a0",
  line_width = 2,
```

```
  opacity = 0.2,
  x_axis = 0L,
  y_axis = 0L,
  layer = NULL
)
```

Arguments

plot	htmlwidget: Existing Cartesian ECharts drawing.
x, y	Numeric: Finite curve coordinates with distinct x values.
lower, upper	Optional Numeric: Paired finite interval bounds at every x.
name	Character: Legend label; a matching observation label links selection.
color	Character: Curve and interval color.
line_width	Numeric (0, Inf): Curve width in pixels.
opacity	Numeric [0, 1]: Interval opacity.
x_axis, y_axis	Integer [0, Inf): Zero-based axis indices.
layer	Optional FitLayer: Complete geometry; do not combine with other settings.

Value

htmlwidget: Drawing with native vector curve and interval layers.

Examples

```
draw_add_fit(draw_scatter(1:3, c(2, 4, 5)), 1:3, c(2.1, 3.8, 5.1), name = "Predicted")
```

---

draw_add_surface *Add a Supplied Surface to a Three-Dimensional Drawing*

Description

Fits are computed explicitly before drawing. Supply predictions from any model on a rectangular grid; fitted R objects and callbacks are not stored in the drawing. The prediction grid may extend the original axis ranges. Add surfaces before composing drawings with draw_panels().

Usage

```
draw_add_surface(
  plot,
  x = NULL,
  y = NULL,
  z = NULL,
  name = "Surface",
  color = "#6ca3a0",
  opacity = 0.35,
  layer = NULL,
  expand = TRUE
)
```

Arguments

plot	htmlwidget: Existing Cartesian ECharts-GL drawing.
x, y	Numeric: Finite, distinct grid coordinates, at least two per axis.
z	Numeric matrix: Predictions, with x along rows and y along columns. NA removes adjacent cells. At least one complete grid cell is required.
name	Character: Legend label; matching names link legend selection.
color	Character: Surface fill color.
opacity	Numeric [0, 1]: Surface opacity.
layer	Optional SurfaceLayer: Complete geometry, supplied alone.
expand	Logical: Extend axis ranges to include surface coordinates. When false, all finite surface coordinates must already lie within them.

Value

htmlwidget: Drawing with native interactive and vector surface layers.

Examples

```
x <- y <- seq(-1, 1, length.out = 5)
draw_add_surface(draw_scatter3d(c(-1, 1), c(-1, 1), c(-1, 1)),
  x, y, outer(x, y, `+`), name = "Predicted")
```

---

draw_annotate *Add References, Bands, and Labels to a Cartesian Drawing*

Description

Adds independent native ECharts layers to an existing drawing. For panels, annotate the children before composition. Annotation coordinates use the existing axes and may extend their automatic ranges; explicit limits remain authoritative. References and bands do not alter data summaries.

Usage

```
draw_annotate(
  plot,
  hline = NULL,
  vline = NULL,
  x = numeric(),
  y = numeric(),
  text = NULL,
  bands = list(),
  color = "#888888",
  line_type = "dashed",
  opacity = 0.15,
  x_axis = 0L,
```

```
  y_axis = 0L,
  layer = NULL
)
```

Arguments

plot	htmlwidget: An ECharts drawing.
hline, vline	Numeric: Horizontal and vertical reference positions.
x, y	Numeric or Character: Coordinates for text annotations.
text	Character: One label per coordinate pair.
bands	List: Numeric length-two x intervals to shade.
color	Character: Annotation color.
line_type	Character {"solid", "dashed", "dotted"}: Reference line style.
opacity	Numeric [0, 1]: Band opacity.
x_axis, y_axis	Integer [0, Inf): Zero-based target axis indices.
layer	Optional AnnotationLayer: Complete geometry. Do not combine with annotation convenience arguments.

Value

htmlwidget: The drawing with an additional vector annotation layer.

Examples

```
draw_annotate(draw_scatter(1:5, c(2, 3, 1, 5, 4)), hline = 3)
```

---

draw_bar *Draw a Bar Chart*

Description

Quick bar chart from x/y data.

Usage

```
draw_bar(
  x,
  y = NULL,
  palette = NULL,
  stack = FALSE,
  horizontal = FALSE,
  xlab = NULL,
  ylab = NULL,
  title = NULL,
  theme = NULL,
```

```
  margins = DEFAULT_MARGINS,
  width = NULL,
  height = NULL,
  element_id = NULL,
  filename = NULL,
  bar_width = NULL,
  order = "input",
  legend_position = "top",
  legend_placement = "outside",
  legend_title = NULL,
  zoom = FALSE
)
```

Arguments

x	Character or Numeric: Category labels, or values when y is omitted. A matrix or two-dimensional table supplies categories in rows and series in columns. A one-dimensional table supplies counts and category names.
y	Numeric or named list: Bar heights.
palette	Optional Character: Bar color or colors. For multiple series, colors are applied per series and recycled as needed. For a single series, a single color styles the whole series; multiple colors are recycled across individual bars. `palette` takes precedence over the theme palette.
stack	Logical: Whether to stack bars.
horizontal	Logical: Whether to draw horizontal bars.
xlab	Optional Character: X-axis title (bottom axis).
ylab	Optional Character: Y-axis title (left axis).
title	Optional Character: Chart title.
theme	Optional Theme: Theme override. The palette inside the theme can be overridden per-chart with the `palette` argument.
margins	Optional Named numeric vector or named list: Plot margins in pixels (or percentage strings) for any of `"top"`, `"right"`, `"bottom"`, `"left"`. Unspecified sides keep echarts' default auto-sizing. See `draw_line()` for details.
width	Optional Character or Numeric: Widget width.
height	Optional Character or Numeric: Widget height.
element_id	Optional Character: Explicit DOM element ID for the widget container. `NULL` lets htmlwidgets generate one.
filename	Optional Character: If provided, save the widget to this file via `save_drawing()`.
bar_width	Optional Numeric `(0, Inf)`: Bar thickness in pixels. NULL lets ECharts size bars from the category spacing.
order	Character `{"input", "increasing", "decreasing"}`: Order categories by their total across series, retaining ties and placing missing totals last.

legend_position
: Character {"top", "bottom", "left", "right", "top-left", "top-right", "bottom-left", "bottom-right"}: Legend anchor. Top/bottom anchors use horizontal rows; left/right anchors use a vertical column. Corner anchors align within the top or bottom row.

legend_placement
: Character {"outside", "inside"}: Relation to the plotting area. Outside placement reserves space for the complete legend; inside placement overlays the data. Neither setting adds a missing legend.

legend_title
: Optional Character: Title drawn with the legend or colorbar: above left/right legends and vertical colorbars, before the first row of top/bottom legends and horizontal colorbars. `NULL` draws no title. A title does not add a missing legend.

zoom
: Logical, DataZoom, or list of DataZoom: Enable zoom on the category axis (the y-axis when `horizontal = TRUE`). `TRUE` adds a slider, Shift+wheel zoom, and drag panning; double-click resets the view. `FALSE` disables zoom. Pass DataZoom objects (or a list of them) for full control.

Value

htmlwidget: Widget object.

Examples

```
draw_bar(x = c("A", "B", "C"), y = c(3, 7, 2), title = "Counts")
```

---

draw_boxplot *Draw a Boxplot with Observation Overlays*

Description

Draw one box per numeric vector, optionally split by groups. Statistics use linear-interpolation quartiles (`stats::quantile(type = 7)`) or Tukey hinges (`stats::fivenum()`). Whiskers extend to observations within the quartiles plus/minus `whisker` times their difference, including the quartiles themselves. `whisker = 0` uses the full observed range. Values strictly outside the fences are outliers; they are never removed from the quartiles.

Usage

```
draw_boxplot(
  x,
  labels = NULL,
  group = NULL,
  horizontal = FALSE,
  palette = NULL,
  fill_alpha = 0.25,
  na_rm = TRUE,
  xlab = NULL,
```

```
  ylab = NULL,
  title = NULL,
  theme = NULL,
  margins = DEFAULT_MARGINS,
  width = NULL,
  height = NULL,
  verbosity = 1L,
  element_id = NULL,
  filename = NULL,
  observation = NULL,
  quartiles = "linear",
  whisker = 1.5,
  boxpoints = "none",
  point_size = 5,
  point_alpha = 0.6,
  point_spread = 0.5,
  legend_position = "top",
  legend_placement = "outside",
  legend_title = NULL,
  geometry = "box",
  bandwidth = NULL,
  adjust = 1,
  density_points = 128L,
  paired = FALSE,
  pair_alpha = 0.35,
  pair_width = 1,
  comparisons = NULL,
  order = "input",
  transform = "none",
  zoom = FALSE
)
```

Arguments

x	Numeric or List: Numeric vectors, one per box; named lists supply category labels. With group, every vector must match its length.
labels	Optional Character: One category label per vector. A single grouped vector uses group labels instead.
group	Optional Atomic vector or single-column data frame: Group identities in first-appearance order. The column name does not add a legend or heading; both input forms produce the same plot.
horizontal	Logical: Draw horizontal boxes.
palette	Optional Character: Box colors, recycled across boxes or groups.
fill_alpha	Numeric [0, 1]: Box fill opacity.
na_rm	Logical: Remove missing values. FALSE rejects missing input.
xlab, ylab	Optional Character: Axis labels.

title	Optional Character: Chart title.
theme	Optional Theme: Theme override. The palette inside the theme can be overridden per-chart with the color argument.
margins	Optional Named numeric vector or named list: Plot margins in pixels (or percentage strings) for any of "top", "right", "bottom", "left" — e.g. c(left = 80, right = 20) or list(left = 80, right = "10%"). Unspecified sides keep echarts' default auto-sizing (outerBoundsMode = "same"), which shrinks the grid to fit axis labels and names with no dead space.
width	Optional Character or Numeric: Widget width.
height	Optional Character or Numeric: Widget height.
verbosity	Integer [0, Inf): Verbosity for missing-value messages.
element_id	Optional Character: Explicit DOM element ID for the widget container. NULL lets htmlwidgets generate one.
filename	Optional Character: If provided, save the widget to this file via save_drawing().
observation	Optional Atomic vector: Observation identifiers in tooltips, aligned to every input vector. Required for pairing; otherwise unset uses original row numbers.
quartiles	Character {"linear", "hinges"}: Quartile convention.
whisker	Numeric [0, Inf): Finite IQR fence multiplier; zero uses full-range whiskers.
boxpoints	Character {"none", "all", "outliers"}: Values to overlay.
point_size	Numeric (0, Inf): Finite point diameter in pixels.
point_alpha	Numeric [0, 1]: Point opacity.
point_spread	Numeric [0, 1]: Fraction of box width occupied by offsets.
legend_position	Character {"top", "bottom", "left", "right", "top-left", "top-right", "bottom-left", "bottom-right"}: Legend anchor. Top/bottom anchors use horizontal rows; left/right anchors use a vertical column. Corner anchors align within the top or bottom row.
legend_placement	Character {"outside", "inside"}: Relation to the plotting area. Outside placement reserves space for the complete legend; inside placement overlays the data. Neither setting adds a missing legend.
legend_title	Optional Character: Title drawn with the legend or colorbar: above left/right legends and vertical colorbars, before the first row of top/bottom legends and horizontal colorbars. NULL draws no title. A title does not add a missing legend.
geometry	Character {"box", "violin", "both"}: Distribution geometry.
bandwidth	Optional Numeric (0, Inf): Finite Gaussian kernel bandwidth.
adjust	Numeric (0, Inf): Finite bandwidth multiplier.
density_points	Integer [16, 4096]: Density evaluation grid size.
paired	Logical: Connect explicit IDs across adjacent categories.
pair_alpha	Numeric [0, 1]: Paired line opacity.
pair_width	Numeric (0, Inf): Finite paired line width in pixels.

comparisons	Optional Data frame: Category endpoints and comparison labels.
order	Character {"input", "mean", "median"}: Keep input order or sort categories by decreasing mean/median after transformation; empty boxes last.
transform	Character {"none", "scale", "minmax"}: Transform each input column before grouping. Scale subtracts the mean and divides by the sample standard deviation; minmax maps the available range to zero through one. Constant columns become zero and missing values remain missing.
zoom	Logical, DataZoom, or list of DataZoom: Enable zoom on the category axis (the y-axis when horizontal = TRUE). TRUE adds a slider, Shift+wheel zoom, and drag panning; double-click resets the view. FALSE disables zoom. Pass DataZoom objects (or a list of them) for full control.

Details

All-NA and empty boxes retain their category without a mark; entirely unavailable input is an error. Missing values are reported in the console according to verbosity and never replaced with zero. Missing group assignments are excluded. Axis titles are centered. The category-axis baseline is shown only when the plotted value range includes zero. Every category label is shown so boxes remain identifiable in narrow plots. Use a horizontal layout or a larger figure for many or long category names. With point overlays, padded range endpoints are unlabeled; interior ticks retain the backend's numeric formatting and plotted values are unchanged. Points retain their exact value coordinate. Their perpendicular offsets use a deterministic base-two sequence in input row order within point_spread times the box width. Zero spread centers every point; coincident points can overlap. No random state is read or changed. Browser and SVG share the same named point renderer, including grouped-box offsets and legend filtering.

Violin densities use a Gaussian kernel with stats::bw.nrd0() unless a numeric bandwidth is supplied, multiplied by adjust. They are evaluated over the observed range and each is scaled to the same maximum width; widths do not encode sample size or comparable density across groups. Singleton and constant samples appear as crossbars, without an invented density. geometry = "both" overlays a narrow box on each violin.

Paired lines join matching explicit observation IDs in adjacent categories only, within groups for multiple grouped vectors. A missing measurement leaves a gap. IDs must be unique within each category/group. Lines meet visible observation points at their deterministic offsets.

Comparison records annotate supplied results; no statistical test or multiplicity adjustment is performed. from and to name categories; label supplies the displayed text. Multiple grouped vectors also require from_group and to_group. Optional finite position values place brackets on the value axis; otherwise brackets stack above the observed geometry. References must identify distinct, nonempty cells. Brackets disappear when either endpoint's group is hidden. Long labels may require a larger figure.

Value

htmlwidget: ECharts boxes with optional vector point overlays.

Examples

```
draw_boxplot(list(Training = c(.8, .9, .85), Test = c(.7, .8, .75)),
```

```
  boxpoints = "all", observation = c("Fold1", "Fold2", "Fold3"))
draw_boxplot(iris["Sepal.Length"], group = iris["Species"])
```

`draw_calibration` *Draw Binary Probability Calibration*

Description

Compare mean predicted probability with observed frequency. Accepts binary labels and probabilities, named lists of paired samples, or individual observation records with `observed`, `probability`, and optional `group`. Named lists are aligned by name. Binary class identity and probability columns follow `draw_roc()`: `positive` names the event, and otherwise the second factor level is selected. Named probability columns identify their classes. Samples must share the same positive class.

Usage

```
draw_calibration(
  true_labels,
  predicted_prob = NULL,
  positive = NULL,
  ...,
  group = NULL,
  legend_position = "top",
  legend_placement = "outside",
  legend_title = NULL,
  theme = NULL,
  width = NULL,
  height = NULL,
  element_id = NULL,
  filename = NULL
)
```

Arguments

`true_labels`	Factor, vector, list, or data frame: Binary labels or observation records.
`predicted_prob`	Optional Numeric vector, matrix, or list: Predicted probabilities.
`positive`	Optional Character: Positive class label.
`...`	Additional named settings for `setup_CalibrationConfig()`.
`group`	Optional Character: Group column for observation records. When omitted, a column named `group` is used if present; explicit NULL pools rows.
`legend_position`	Character {"top", "bottom", "left", "right", "top-left", "top-right", "bottom-left", "bottom-right"}: Legend anchor. Top/bottom anchors use horizontal rows; left/right anchors use a vertical column. Corner anchors align within the top or bottom row.

`legend_placement`
: Character {"outside", "inside"}: Relation to the plotting area. Outside placement reserves space for the complete legend; inside placement overlays the data. Neither setting adds a missing legend.

`legend_title`
: Optional Character: Title drawn with the legend or colorbar: above left/right legends and vertical colorbars, before the first row of top/bottom legends and horizontal colorbars. `NULL` draws no title. A title does not add a missing legend.

`theme`
: Optional Theme: Chart theme.

`width, height`
: Optional Numeric or Character: Widget dimensions.

`element_id`
: Optional Character: HTML element identifier.

`filename`
: Optional Character: Static output path, currently SVG.

Value

An ECharts htmlwidget with vector SVG export.

Statistical semantics

Bins are computed independently within each group. Equidistant bins partition the unit interval. Quantile bins use linear interpolation at positions `(n - 1) * p + 1` in the sorted probabilities (R quantile type 7). Repeated boundaries are collapsed without splitting tied scores. Intervals include their lower boundary and exclude their upper boundary, except that the final interval includes its upper boundary. Constant probabilities form one bin. Empty bins are omitted; lines join the remaining bin means. Each point is the mean predicted probability and mean observed outcome in its bin. The Brier score is the mean squared probability error across all complete observations, before binning. It is not an average of bin errors. Missing pairs are removed together when `na_rm` is true; invalid finite ranges and infinite values are always rejected. A group with no complete observations is rejected. The probability rug uses those same complete observations and sits just inside the lower plotting edge. Both axes span zero to one. Display precision does not round plotted values. A one-bin curve remains visible as a point even in lines-only mode.

Examples

```
labels <- factor(c("no", "yes", "no", "yes"), levels = c("no", "yes"))
draw_calibration(labels, c(.1, .8, .4, .9), n_bins = 2L)
```

`draw_choropleth` *Draw a Choropleth Map*

Description

Render a choropleth map with MapLibre GL: regions shaded by a value, joined to vendored administrative boundaries (countries by ISO-A3, US states / counties by FIPS). No basemap tiles – boundaries render on the themed background, with light/dark auto-detection like the rest of `draw_*`.

Usage

```
draw_choropleth(
  x,
  location,
  value,
  resolution = c("country", "state", "county"),
  tooltip = NULL,
  value_label = NULL,
  classification = "quantile",
  colormap = "blues",
  num_classes = 5,
  opacity = 1,
  show_boundaries = TRUE,
  outline_width = 0.2,
  show_legend = TRUE,
  legend_position = "bottom-right",
  tooltip_position = "top-right",
  title = NULL,
  theme = NULL,
  width = NULL,
  height = NULL,
  element_id = NULL,
  filename = NULL
)
```

Arguments

x	Data frame: One row per region.
location	Character: Name of the column holding the location key (FIPS / ISO / name).
value	Character: Name of the numeric column to color by.
resolution	Character {"country", "state", "county"}: Administrative resolution selecting the geometry and the join key.
tooltip	Optional Character: Names of extra columns to show in the hover tooltip, in order.
value_label	Optional Character: Label for the value column in the legend / tooltip; defaults to value.
classification	Character {"quantile", "equal", "jenks"}: Class-break method. "quantile" (equal counts), "equal" (equal intervals), or "jenks" (natural breaks).
colormap	Character: Color ramp. One of the sequential schemes "blues", "viridis", "ylorrd", "greens", "magma", or the diverging schemes "rdbu", "rdylgn", "spectral", "brbg".
num_classes	Numeric [2, 12]: Number of color classes.
opacity	Numeric [0, 1]: Region fill opacity.
show_boundaries	Logical: Whether to draw region outlines.

outline_width	Numeric [0, Inf): Region outline width in pixels.
show_legend	Logical: Whether to show the legend.
legend_position	Character {"top-left", "top-right", "bottom-left", "bottom-right"}: Legend corner.
tooltip_position	Character {"top-left", "top-right", "bottom-left", "bottom-right"}: Hover tooltip corner.
title	Optional Character: Title (currently surfaced via the value label; reserved for future use).
theme	Optional Theme, list, or NA: Theme override. NULL enables light/dark auto-detection (matching draw()).
width	Optional Character or Numeric: Widget width.
height	Optional Character or Numeric: Widget height.
element_id	Optional Character: Explicit element ID.
filename	Optional Character: Static export of map widgets is not yet supported. Supplying a path raises an error.

Details

The location column is matched to the geometry in the renderer, which accepts ISO-A3 / ISO-A2 / country code for countries, and FIPS / postal abbreviation / full name for US states (5-digit FIPS for counties). A join report in the corner surfaces any unmatched keys.

Value

htmlwidget.

Examples

```
# Country choropleth from ISO-A3 codes
df <- data.frame(
  iso = c("USA", "CAN", "MEX", "BRA", "FRA"),
  gdp = c(25.5, 2.1, 1.4, 1.9, 2.9)
)
draw_choropleth(df, location = "iso", value = "gdp")

# US states by postal abbreviation, natural-breaks classification
states <- data.frame(
  st = state.abb,
  pop = as.numeric(state.x77[, "Population"])
)
draw_choropleth(
  states,
  location = "st",
  value = "pop",
  resolution = "state",
  classification = "jenks",
```

```
  colormap = "viridis"
)
```

`draw_confusion` *Draw Confusion Counts and Classification Summaries*

Description

Accepts a square count matrix, a named list of count matrices, long frequency records (`reference`, `predicted`, `n`, optional `panel`), or paired reference and predicted label vectors. Model packages can add support for metrics objects through `confusion_input()`. Matrix columns are aligned by label identity; their physical order need not match rows. All panels share one class order.

Usage

```
draw_confusion(
  x,
  y = NULL,
  classes = NULL,
  ...,
  theme = NULL,
  width = NULL,
  height = NULL,
  element_id = NULL,
  filename = NULL
)
```

Arguments

`x`	Matrix, table, data frame, named list, metrics object, or vector: Counts or reference labels.
`y`	Optional vector: Predicted labels when x contains reference labels.
`classes`	Optional Character vector: Ordered class labels shared by every panel; unset preserves first appearance.
`...`	Additional named settings for `setup_ConfusionConfig()`.
`theme`	Optional Theme: Chart theme. Color fades and marginal backgrounds follow the active theme unless overridden in the config.
`width, height`	Optional Numeric or Character: Widget dimensions.
`element_id`	Optional Character: HTML element identifier.
`filename`	Optional Character: Static output path, currently SVG.

Value

An ECharts htmlwidget containing every requested panel.

Statistical semantics

Color intensity is the fraction within each reference row; labels show raw counts. Sensitivity, specificity, PPV, and NPV are one-versus-rest rates for each named class. Balanced accuracy is mean class recall, including every declared class. Zero denominators produce NA, never zero; consequently balanced accuracy is NA if any declared class has no reference observations. Summaries of pooled resamples describe pooled predictions, not mean fold performance. Repeated observations are counted each time they appear.

Examples

```
draw_confusion(factor(c("yes", "yes", "no")), c("yes", "no", "no"))
draw_confusion(matrix(c(8, 2, 1, 9), 2,
  dimnames = list(c("yes", "no"), c("yes", "no"))))
```

---

draw_density *Draw a Density Plot*

Description

Kernel density estimation plot from numeric data, with optional grouping for multiple traces. A list input creates one density trace per vector when ungrouped, or one trace per variable/group combination when group is supplied. Curves use stats::density() with its bandwidth-scaled kernels and evaluation range extending three bandwidths beyond the data. Empty samples retain their identity without a curve. Singleton and constant samples require an explicit bandwidth; no spread is inferred for them. Missing groups are excluded, and infinite/nonnumeric observations rejected.

Usage

```
draw_density(
  x,
  group = NULL,
  n = 512,
  bw = "nrd0",
  na_rm = TRUE,
  palette = NULL,
  xlab = NULL,
  ylab = NULL,
  title = NULL,
  theme = NULL,
  margins = DEFAULT_MARGINS,
  width = NULL,
  height = NULL,
  verbosity = 1L,
  element_id = NULL,
  filename = NULL,
  legend_position = "top",
```

```
  legend_placement = "outside",
  legend_title = NULL,
  bandwidth = NULL,
  kernel = "gaussian",
  adjust = 1,
  fill_alpha = 0.25,
  mode = "overlap",
  order = "input",
  zoom = FALSE
)
```

Arguments

x	Numeric or list: Values used for density estimation. An ungrouped list creates one density trace per element; with group, each list element is split by group into separate traces.
group	Optional Atomic vector or single-column data frame: Grouping variable for multiple density traces.
n	Integer [2, Inf): Number of density evaluation points.
bw	Character or Numeric: Bandwidth selector ("nrd0", "nrd", "ucv", "bcv", "SJ", "SJ-ste", "SJ-dpi"). Numeric input is normalized to bandwidth by the setup function.
na_rm	Logical: Whether to remove NA values before computing densities.
palette	Optional Character: Series colors, overriding the theme palette for this chart. NULL uses the theme's.
xlab	Optional Character: X-axis title.
ylab	Optional Character: Y-axis title.
title	Optional Character: Chart title.
theme	Optional Theme: Theme override.
margins	Optional Named numeric vector or named list: Plot margins in pixels (or percentage strings) for any of "top", "right", "bottom", "left". Unspecified sides keep echarts' default auto-sizing. See draw_line() for details.
width	Optional Character or Numeric: Widget width.
height	Optional Character or Numeric: Widget height.
verbosity	Integer [0, Inf): Verbosity level for removed-NA messages.
element_id	Optional Character: Explicit DOM element ID for the widget container. NULL lets htmlwidgets generate one.
filename	Optional Character: If provided, save the widget to this file via save_drawing().
legend_position	Character {"top", "bottom", "left", "right", "top-left", "top-right", "bottom-left", "bottom-right"}: Legend anchor. Top/bottom anchors use horizontal rows; left/right anchors use a vertical column. Corner anchors align within the top or bottom row.

`legend_placement`
: Character {"outside", "inside"}: Relation to the plotting area. Outside placement reserves space for the complete legend; inside placement overlays the data. Neither setting adds a missing legend.

`legend_title`
: Optional Character: Title drawn with the legend or colorbar: above left/right legends and vertical colorbars, before the first row of top/bottom legends and horizontal colorbars. `NULL` draws no title. A title does not add a missing legend.

`bandwidth`
: Optional Numeric `(0, Inf)`: Explicit finite smoothing bandwidth in measurement units, overriding the selector.

`kernel`
: Character: One of `"gaussian"`, `"epanechnikov"`, `"rectangular"`, `"triangular"`, `"biweight"`, `"cosine"`, or `"optcosine"`.

`adjust`
: Numeric `(0, Inf)`: Finite bandwidth multiplier.

`fill_alpha`
: Numeric `[0, 1]`: Distribution fill opacity.

`mode`
: Character `{"overlap", "ridge"}`: Overlay samples or align rows on common x and y scales. Ridge rows use sample names in place of a legend. Increase figure height for many rows.

`order`
: Character `{"input", "mean", "median"}`: Sample order. Summary orders are decreasing, retain ties in input order, and place empty samples last.

`zoom`
: Logical, DataZoom, or list of DataZoom: Enable x-axis zoom. `TRUE` adds a slider below the axis title, Shift+wheel zoom, and drag panning; in ridge mode, every row zooms together and no slider is drawn. Double-click resets the view. `FALSE` disables zoom. Pass DataZoom objects (or a list of them) for full control.

Value

htmlwidget: Widget object.

Examples

```
draw_density(iris[["Sepal.Length"]], xlab = "Sepal length")
```

---

draw_fit *Draw True Versus Predicted Values*

Description

Draw paired observations with a fitted line, standard-error band, and an identity line. The plotting box is square and both axes share an interval. Named lists support samples of different lengths, with one fit per sample.

Usage

```
draw_fit(
  x,
  y,
  group = NULL,
  fit = "glm",
  se = TRUE,
  se_times = 1.96,
  rsq = TRUE,
  diagonal = TRUE,
  diagonal_color = NULL,
  square = TRUE,
  equal_axes = TRUE,
  xlab = "True",
  ylab = "Predicted",
  ...,
  legend_position = "top",
  legend_placement = "outside",
  legend_title = NULL
)
```

Arguments

x	Numeric vector, single-column matrix, or list: True values.
y	Numeric vector, single-column matrix, or list: Predicted values.
group	Optional Atomic vector or single-column data frame: Grouping variable for multiple series.
fit	Optional Character: Model fitted to y on x and drawn as a line. "glm" uses stats::glm() and "gam" uses mgcv::gam(). Any other name, matched case-insensitively, is an rtemis supervised learning algorithm, such as "LINAD", "CART" or "LightGBM", trained with its default hyperparameters; this requires the rtemis package (>= 1.4.1). NULL disables fitting. The fitted line and standard-error band are computed per group when group is provided.
se	Logical: Whether to show the standard-error band. Algorithms without standard errors (every rtemis algorithm except GLM, GAM and a few others) draw the line alone.
se_times	Numeric [0, Inf): Standard-error multiplier for the band.
rsq	Logical: Include the fitted model's R-squared in series labels. This describes the overlay fit, not predictive performance against the identity line. Constant responses have undefined R-squared, labeled NA.
diagonal	Logical: Draw a dashed identity line within the axis limits.
diagonal_color	Optional Character: Identity-line color.
square	Logical: If TRUE, draw the plotting box square – equal height and width in pixels, excluding axis labels and margins.

`equal_axes` Logical: If TRUE, give one data unit the same size in pixels on both axes. Set both for a plot that is square *and* to scale, such as a true-versus-predicted plot whose identity line should run at 45 degrees; the two axes are then made to span the same interval. See Details.

`xlab` Optional Character: X-axis title.

`ylab` Optional Character: Y-axis title.

`...` Additional arguments to `draw_scatter()`, including `theme`, `palette`, limits, margins, dimensions, and `filename`.

`legend_position`
Character {"top", "bottom", "left", "right", "top-left", "top-right", "bottom-left", "bottom-right"}: Legend anchor. Top/bottom anchors use horizontal rows; left/right anchors use a vertical column. Corner anchors align within the top or bottom row.

`legend_placement`
Character {"outside", "inside"}: Relation to the plotting area. Outside placement reserves space for the complete legend; inside placement overlays the data. Neither setting adds a missing legend.

`legend_title` Optional Character: Title drawn with the legend or colorbar: above left/right legends and vertical colorbars, before the first row of top/bottom legends and horizontal colorbars. `NULL` draws no title. A title does not add a missing legend.

Details

The overlay uses the same implementation as `draw_scatter()` and ScatterConfig. The fit is GLM, GAM, or an rtemis learner named in `fit`; `fit = NULL` disables it. Missing pairs and observations with missing groups are dropped together. Infinite values, unequal pair lengths, and multiple response columns are rejected. List names are matched before drawing; unnamed lists are paired by position. Paired `NULL` entries in lists represent unavailable sets and are omitted.

The equivalent portable configuration is a ScatterConfig binding true, predicted, and sample columns, with `fit`, `se`, `rsq`, `diagonal`, `square`, and `equal_axes` set explicitly. Model objects are not needed to render it.

Value

htmlwidget: ECharts drawing.

Examples

```
draw_fit(1:5, c(1.2, 1.8, 3.3, 3.8, 5.2))
draw_fit(
  list(Training = 1:5, Test = 2:5),
  list(Training = c(1.2, 1.8, 3.3, 3.8, 5.2), Test = c(2.2, 2.7, 4.3, 4.8))
)
```

draw_gantt *Draw a Timeline / Gantt Chart*

Description

Draw a timeline (Gantt) chart: one horizontal bar per task, positioned by `start` and `end` on a value or time x-axis and grouped into rows by `label` on the y-axis. Implemented as an ECharts `custom` series (ECharts has no native Gantt series).

Usage

```
draw_gantt(
  tasks,
  group = NULL,
  axis_type = "value",
  bar_height = 0.6,
  bar_radius = 0,
  guides = TRUE,
  zoom = TRUE,
  tooltip = NULL,
  border = NULL,
  border_color = "#E53935",
  border_width = 1.5,
  xlab = NULL,
  title = NULL,
  palette = NULL,
  theme = NULL,
  width = NULL,
  height = NULL,
  element_id = NULL,
  filename = NULL,
  legend_position = "top",
  legend_placement = "outside",
  legend_title = NULL
)
```

Arguments

`tasks`	Tabular data (data.frame, data.table, or tibble): One row per task bar. Must contain columns `label` (row / category), `start`, and `end`. Repeated `label` values place multiple bars on the same row.
`group`	Optional Character: Name of a column in `tasks` whose values color the bars and produce a legend. When `NULL`, all bars share one color.
`axis_type`	Character {"value", "time"}: Type of the x-axis. Use `"value"` for numeric offsets (e.g. milliseconds from start) and `"time"` for absolute timestamps; `POSIXct`/`Date` `start`/`end` columns are converted to epoch milliseconds automatically.

bar_height — Numeric (0, 1]: Bar thickness as a fraction of one category band.

bar_radius — Numeric [0, Inf): Bar corner radius in pixels.

guides — Logical: If TRUE, show an interactive axis pointer – a guide line that follows the mouse and labels the time value on the x-axis.

zoom — Logical: If TRUE, enable interactive zoom – Shift+wheel to zoom and drag to pan on both the time and row axes (inside dataZoom), plus a top-right toolbox with box-zoom, undo, and reset controls.

tooltip — Optional Character: Name of a column in tasks to show as the tooltip text for each bar. When NULL, a default label: start - end tooltip is shown.

border — Optional Character: Name of a logical column in tasks; bars whose value is TRUE get an outline (in border_color) without changing their fill – e.g. to flag failures while the fill still encodes the group.

border_color — Character: Outline color for bars flagged by border.

border_width — Numeric [0, Inf): Outline width in pixels.

xlab — Optional Character: x-axis label.

title — Optional Character: Chart title.

palette — Optional Character: Color palette as a single color or character vector overriding the theme palette. Groups are colored in order.

theme — Optional Theme: Theme override.

width — Optional Character or Numeric: Widget width.

height — Optional Character or Numeric: Widget height.

element_id — Optional Character: Explicit DOM element ID for the widget container. NULL lets htmlwidgets generate one.

filename — Optional Character: If provided, save the widget to this file via save_drawing().

legend_position — Character {"top", "bottom", "left", "right", "top-left", "top-right", "bottom-left", "bottom-right"}: Legend anchor. Top/bottom anchors use horizontal rows; left/right anchors use a vertical column. Corner anchors align within the top or bottom row.

legend_placement — Character {"outside", "inside"}: Relation to the plotting area. Outside placement reserves space for the complete legend; inside placement overlays the data. Neither setting adds a missing legend.

legend_title — Optional Character: Title drawn with the legend or colorbar: above left/right legends and vertical colorbars, before the first row of top/bottom legends and horizontal colorbars. NULL draws no title. A title does not add a missing legend.

Details

The legend uses the shared position/placement controls and reserves enough room for the time axis. Tick density follows the available plotting width, and long task labels are truncated. Default bar tooltips retain the full task text. Browser resizing, panel figures, and SVG export share this layout behavior. Increase height when a timeline contains many task rows.

Value

htmlwidget: Widget object.

Author(s)

EDG

Examples

```
tasks <- data.frame(
  label = c("load", "clean", "train", "predict"),
  start = c(0, 12, 30, 95),
  end = c(12, 30, 95, 100),
  status = c("ok", "ok", "ok", "error")
)
draw_gantt(tasks, group = "status")
```

---

draw_graph *Render a GraphModel as a Sigma.js htmlwidget*

Description

Low-level renderer: takes a GraphModel (or a plain list with `nodes`, `edges`, `directed`) and produces an interactive network htmlwidget backed by Sigma.js + graphology. Most users call `draw_network()` instead.

Usage

```
draw_graph(
  model,
  layout = "force",
  node_size = 10,
  edge_scale = 3,
  node_opacity = 0.95,
  edge_opacity = 0.4,
  show_labels = TRUE,
  scale_by_degree = TRUE,
  color_by_group = FALSE,
  resolution = 1,
  blend_edges = FALSE,
  palette = palette_colors(rtemis_colors),
  node_color = rtemis_colors[["teal"]],
  positive_color = NULL,
  negative_color = NULL,
  title = NULL,
  theme = NULL,
  width = NULL,
```

```
  height = NULL,
  element_id = NULL,
  filename = NULL
)
```

Arguments

model	GraphModel or named list: The graph to render.
layout	Character {"force", "circular", "circlepack", "random"}: Layout algorithm. "force" is ForceAtlas2.
node_size	Numeric [0, Inf): Base node radius in screen pixels.
edge_scale	Numeric [0, Inf): Multiplier mapping normalized edge weight to stroke width.
node_opacity	Numeric [0, 1]: Node fill opacity.
edge_opacity	Numeric [0, 1]: Edge stroke opacity.
show_labels	Logical: Whether to render node labels.
scale_by_degree	Logical: Scale each node's radius by its value (weighted degree); when FALSE all nodes use the base size.
color_by_group	Logical: Color nodes by detected community (Louvain) instead of a single hue.
resolution	Numeric [0, Inf): Louvain resolution; higher yields more, smaller communities.
blend_edges	Logical: Color each edge as the blend of its two endpoint node colors instead of by sign.
palette	Character: Categorical colors for communities.
node_color	Character: Single node color used when color_by_group is FALSE.
positive_color	Optional Character: Edge color for positive-sign edges. NULL uses the theme's positive color.
negative_color	Optional Character: Edge color for negative-sign edges. NULL uses the theme's negative color.
title	Optional Character: Title shown above the network.
theme	Optional Theme, list, or NA: Theme override. NULL enables light/dark auto-detection (matching draw()).
width	Optional Character or Numeric: Widget width.
height	Optional Character or Numeric: Widget height.
element_id	Optional Character: Explicit element ID.
filename	Optional Character: Static export of network widgets is not yet supported. Supplying a path raises an error.

Value

htmlwidget.

Examples

```
model <- GraphModel(
  nodes = list(GraphNode(id = "a"), GraphNode(id = "b")),
  edges = list(GraphEdge(source = "a", target = "b")),
  directed = FALSE
)
draw_graph(model)
```

`draw_harmonograph` *Draw a Harmonograph*

Description

Draws the curve traced by a harmonograph: two damped pendulums move the pen along x and two along y.

Usage

```
draw_harmonograph(
  frequency = NULL,
  damping = NULL,
  phase = NULL,
  t = seq(1, 500, by = 0.01),
  seed = NULL,
  color = NULL,
  alpha = 0.2,
  line_width = 1,
  pad = DEFAULT_PAD,
  title = NULL,
  theme = NULL,
  margins = DEFAULT_MARGINS,
  width = NULL,
  height = NULL,
  element_id = NULL,
  filename = NULL
)
```

Arguments

`frequency`	Optional Numeric: Pendulum frequencies, length 4. The first two move the pen along x, the last two along y.
`damping`	Optional Numeric `[0, Inf)`: Pendulum damping rates, length 4.
`phase`	Optional Numeric: Pendulum phases in radians, length 4.
`t`	Numeric: Time points at which the curve is evaluated.
`seed`	Optional Integer: Seed for the random parameters. The caller's random number generator state is restored on exit. `NULL` uses the current state.

color	Optional Character: Line color. NULL uses the theme palette's first color.
alpha	Numeric [0, 1]: Line opacity.
line_width	Numeric (0, Inf): Line width in pixels.
pad	Numeric [0, Inf): Fraction of the curve's larger range added on each side.
title	Optional Character: Chart title.
theme	Optional Theme: Theme override. NULL follows the light or dark appearance of the viewer.
margins	Optional Named numeric vector or named list: Plot margins in pixels for any of "top", "right", "bottom", "left".
width	Optional Character or Numeric: Widget width.
height	Optional Character or Numeric: Widget height.
element_id	Optional Character: Explicit DOM element ID for the widget container. NULL lets htmlwidgets generate one.
filename	Optional Character: If provided, save the widget to this file via save_drawing().

Details

The curve is

$$x(t) = e^{-d_1 t}\sin(f_1 t + p_1) + e^{-d_2 t}\sin(f_2 t + p_2)$$

$$y(t) = e^{-d_3 t}\sin(f_3 t + p_3) + e^{-d_4 t}\sin(f_4 t + p_4)$$

with frequencies f, damping rates d and phases p.

Parameters left NULL are drawn at random: each frequency is 2 or 3, perturbed by base::jitter(); damping rates are uniform on [0, 0.01]; phases are uniform on [0, pi]. All 12 parameters are drawn in that order and supplied vectors replace their drawn counterparts, so a fixed seed keeps the remaining parameters fixed whichever vectors are supplied. A given seed yields the same parameters as mplot3_harmonograph() in rtemis 0.98. Seeds to try include 9, 17, 26, 202 and 208.

The line follows the theme: color = NULL uses the first palette color, on the theme background. The curve is drawn as up to 500 consecutive segments, and the opacity of overlapping segments accumulates, so dense regions of the curve appear brighter.

The curve formulation follows the Fronkonstin blog post "Beautiful Curves: The Harmonograph" (2014).

Value

htmlwidget: Widget object.

Author(s)

EDG

Examples

```
draw_harmonograph(seed = 9)
draw_harmonograph(
  frequency = c(2, 3, 3, 2),
  damping = rep(0.004, 4),
  phase = c(0, pi / 2, pi / 4, 0),
  alpha = 0.5
)
```

---

draw_heatmap *Draw a Heatmap*

Description

Quick heatmap from a numeric matrix. Designed to handle all common use cases including correlation matrices (square cells, diverging color scale), general rectangular heatmaps, and clustered heatmaps.

Usage

```
draw_heatmap(
  x,
  row_names = NULL,
  col_names = NULL,
  triangle = NULL,
  cluster_rows = FALSE,
  cluster_cols = FALSE,
  dist_method = "euclidean",
  hclust_method = "complete",
  show_row_dendro = TRUE,
  show_col_dendro = TRUE,
  dendro_row_width = 60,
  dendro_col_height = 60,
  dendro_color = NULL,
  dendro_uniform = FALSE,
  dendro_row_side = "right",
  dendro_col_side = "top",
  square_cells = NULL,
  colormap = NULL,
  zlim = NULL,
  show_values = FALSE,
  value_digits = 2L,
  show_colorbar = TRUE,
  colorbar_orient = "vertical",
  title = NULL,
  theme = NULL,
  margins = NULL,
```

```
  width = NULL,
  height = NULL,
  element_id = NULL,
  filename = NULL,
  legend_position = if (colorbar_orient == "horizontal") "bottom" else "right",
  legend_placement = "outside",
  legend_title = NULL,
  row_tree = NULL,
  col_tree = NULL,
  cell_notes = NULL,
  row_colors = NULL,
  col_colors = NULL,
  show_notes = FALSE,
  row_cut = NULL,
  col_cut = NULL
)
```

Arguments

x	Numeric matrix: Input data. Rows map to y-axis categories and columns to x-axis categories.
row_names	Optional Character: Row labels. Defaults to `rownames(x)`, or `"1"`, `"2"`, ... when row names are absent.
col_names	Optional Character: Column labels. Defaults to `colnames(x)`.
triangle	Optional Character {"upper", "lower"}: Mask one triangle of the matrix to `NA`. `"upper"` keeps only the upper triangle; `"lower"` keeps only the lower triangle. The diagonal is always masked, as it is uninformative for symmetric matrices (e.g. always 1 for correlations).
cluster_rows	Logical: Whether to reorder rows via hierarchical clustering.
cluster_cols	Logical: Whether to reorder columns via hierarchical clustering.
dist_method	Character: Distance method passed to `stats::dist()`. Common values: `"euclidean"`, `"manhattan"`.
hclust_method	Character: Linkage method passed to `stats::hclust()`. Common values: `"complete"`, `"ward.D2"`, `"average"`.
show_row_dendro	Logical: Whether to render the row dendrogram panel when `cluster_rows = TRUE`. Set to `FALSE` to reorder rows but suppress the visual dendrogram.
show_col_dendro	Logical: Whether to render the col dendrogram panel when `cluster_cols = TRUE`.
dendro_row_width	Optional Numeric `[1, Inf)`: Pixel width of the row dendrogram panel (between row labels and the heatmap grid).
dendro_col_height	Optional Numeric `[1, Inf)`: Pixel height of the col dendrogram panel (between the chart title and the heatmap grid).

dendro_color Optional Character: Stroke color for dendrogram branch lines. NULL (default) uses a semi-transparent gray ("#99999988"), which works in both light and dark themes.

dendro_uniform Logical: Whether to render dendrograms with uniform level heights — each merge step occupies equal visual space — rather than heights proportional to actual merge distances. FALSE (default) preserves merge distances in the visual scaling.

dendro_row_side
Character {"right", "left"}: Side of the heatmap on which the row dendrogram is placed. "right" (default) keeps row labels on the left with the dendrogram on the right for a clean, symmetric layout.

dendro_col_side
Character {"top", "bottom"}: Side of the heatmap on which the column dendrogram is placed. "top" (default) places it above the heatmap. "bottom" automatically moves column labels to the top.

square_cells Optional Logical: Whether to compute widget dimensions so cells are square in the browser and SVG exports. In panels and exports, the matrix and dendrograms fit within the allocated canvas. NULL (default) enables this automatically for square matrices (e.g. correlation matrices). When TRUE, both width and height are calculated from the number of cells; supply explicit width/height to override.

colormap Optional Character: Color palette — a vector of 2 or more colors defining the continuous color scale from zlim[1] to zlim[2]. When NULL (default) a diverging negative-background-positive palette in the theme's sign colors is used when data spans zero (with the theme's background pinned exactly at 0, even for asymmetric ranges), otherwise the half of that scale matching the sign of the data: negative to background when no value is positive, background to positive when no value is negative. Two variants (light / dark) are computed automatically and the JS binding selects the correct one based on the active theme.

zlim Optional Numeric: Length-2 vector c(min, max) for the color scale. Defaults to the observed data range. For correlation matrices, c(-1, 1) is recommended.

show_values Logical: Whether to print the cell value as a label inside each cell.

value_digits Integer: Decimal places used in cell labels and the tooltip.

show_colorbar Logical: Whether to display the continuous color-scale bar.

colorbar_orient
Character {"vertical", "horizontal"}: Orientation of the color bar.

title Optional Character: Chart title.

theme Optional Theme: Theme override. NULL auto-detects light/dark mode.

margins Optional Named numeric vector or named list: Override the auto-computed heatmap plot-area margins in pixels (or percentage strings) for any of "top", "right", "bottom", "left" — e.g. c(left = 200) to widen the left gutter for long row labels. Unspecified sides use the values computed from label lengths, title, colorbar, and dendrogram panels. See draw_line() for the general convention.

width Optional Character or Numeric: Widget width.

`height` Optional Character or Numeric: Widget height.

`element_id` Optional Character: Explicit DOM element ID for the widget container. `NULL` lets htmlwidgets generate one.

`filename` Optional Character: If provided, save the widget to this file via `save_drawing()`.

`legend_position` Character {"top", "bottom", "left", "right", "top-left", "top-right", "bottom-left", "bottom-right"}: Legend anchor. Top/bottom anchors use horizontal rows; left/right anchors use a vertical column. Corner anchors align within the top or bottom row.

`legend_placement` Character {"outside", "inside"}: Relation to the plotting area. Outside placement reserves space for the complete legend; inside placement overlays the data. Neither setting adds a missing legend.

`legend_title` Optional Character: Title drawn with the legend or colorbar: above left/right legends and vertical colorbars, before the first row of top/bottom legends and horizontal colorbars. `NULL` draws no title. A title does not add a missing legend.

`row_tree`, `col_tree` Optional hclust, dendrogram or List: Supplied trees. A plain tree record contains `merge`, `height`, `order` and optional `labels`. Labeled leaves match original matrix dimnames; unlabeled trees are positional. Supplied trees determine order and take precedence over computed clustering. Triangle masking is unavailable with supplied trees.

`cell_notes` Optional Character matrix: Literal cell notes for hover. Named dimensions align to original matrix identities; otherwise positional.

`row_colors`, `col_colors` Optional Character vector, matrix or Data frame: R colors, one row per matrix row/column and one column per annotation track. Row names, when supplied, match the original matrix identities.

`show_notes` Logical: Print cell notes instead of numeric value labels. Long notes truncate to the cell width; hover retains the full text.

`row_cut`, `col_cut` Optional Integer: Number of clusters for branch coloring. A cut requires a supplied or computed tree. Cross-cluster branches stay gray (or `dendro_color`); within-cluster branches share a deterministic color.

Details

Color encoding is driven by a continuous VisualMap component. Hierarchical clustering is performed in R via `hclust()`; rows and columns are reordered accordingly. When `cluster_rows` or `cluster_cols` is `TRUE` (and the corresponding `show_*_dendro` flag is not `FALSE`), a dendrogram panel is rendered alongside the heatmap as an ECharts custom series.

Value

htmlwidget: Widget object.

Examples

```
draw_heatmap(cor(mtcars[, 1:5]), title = "Correlation")
```

`draw_histogram` *Draw a Histogram*

Description

Histogram from numeric data, with optional grouping for multiple traces. Bins are computed using `graphics::hist()` with consistent break points across groups. Rectangles span their actual numeric intervals, including unequal widths. Grouped samples overlap with translucent fills. Intervals are right-closed, with the lowest edge included, using `hist()` boundary tolerance. Missing groups are excluded; empty samples have zero counts.

Usage

```
draw_histogram(
  x,
  group = NULL,
  breaks = "Sturges",
  palette = NULL,
  xlab = NULL,
  ylab = NULL,
  title = NULL,
  theme = NULL,
  margins = DEFAULT_MARGINS,
  width = NULL,
  height = NULL,
  element_id = NULL,
  filename = NULL,
  legend_position = "top",
  legend_placement = "outside",
  legend_title = NULL,
  bins = NULL,
  bin_edges = NULL,
  normalization = "count",
  density = FALSE,
  n = 512L,
  bw = "nrd0",
  bandwidth = NULL,
  kernel = "gaussian",
  adjust = 1,
  fill_alpha = NULL,
  border_alpha = 1,
  na_rm = TRUE,
  verbosity = 1L,
  mode = "overlap",
```

```
  order = "input",
  bin_stat = "count",
  bar_mode = "overlay",
  zoom = FALSE
)
```

Arguments

x	Numeric or List: Values or named numeric vectors to bin.
group	Optional Atomic vector or single-column data frame: Grouping variable for multiple series.
breaks	Numeric, Character, or Numeric vector: Binning method. A single number (number of bins), a character string naming an algorithm (e.g. "Sturges", "Scott", "FD"), or a numeric vector of break points. Numeric forms are normalized to bins or bin_edges by the setup function. Bin counts are suggestions to hist().
palette	Optional Character: Series colors, overriding the theme palette for this chart. NULL uses the theme's.
xlab	Optional Character: X-axis title.
ylab	Optional Character: Y-axis title.
title	Optional Character: Chart title.
theme	Optional Theme: Theme override.
margins	Optional Named numeric vector or named list: Plot margins in pixels (or percentage strings) for any of "top", "right", "bottom", "left". Unspecified sides keep echarts' default auto-sizing. See draw_line() for details.
width	Optional Character or Numeric: Widget width.
height	Optional Character or Numeric: Widget height.
element_id	Optional Character: Explicit DOM element ID for the widget container. NULL lets htmlwidgets generate one.
filename	Optional Character: If provided, save the widget to this file via save_drawing().
legend_position	Character {"top", "bottom", "left", "right", "top-left", "top-right", "bottom-left", "bottom-right"}: Legend anchor. Top/bottom anchors use horizontal rows; left/right anchors use a vertical column. Corner anchors align within the top or bottom row.
legend_placement	Character {"outside", "inside"}: Relation to the plotting area. Outside placement reserves space for the complete legend; inside placement overlays the data. Neither setting adds a missing legend.
legend_title	Optional Character: Title drawn with the legend or colorbar: above left/right legends and vertical colorbars, before the first row of top/bottom legends and horizontal colorbars. NULL draws no title. A title does not add a missing legend.
bins	Optional Integer [1, 1000000]: Suggested bin count.

bin_edges	Optional Numeric vector: Finite, strictly increasing edges spanning every retained observation. Overrides breaks; excludes bins.
normalization	Character: "count", "probability", "percent", "density", or "count_density".
density	Logical: Overlay a kernel density in histogram units.
n	Integer [2, Inf): Number of density evaluation points.
bw	Character or Numeric: Bandwidth selector ("nrd0", "nrd", "ucv", "bcv", "SJ", "SJ-ste", "SJ-dpi"). Numeric input is normalized to bandwidth by the setup function.
bandwidth	Optional Numeric (0, Inf): Explicit finite smoothing bandwidth in measurement units, overriding the selector.
kernel	Character: One of "gaussian", "epanechnikov", "rectangular", "triangular", "biweight", "cosine", or "optcosine".
adjust	Numeric (0, Inf): Finite bandwidth multiplier.
fill_alpha	Optional Numeric [0, 1]: Bar fill opacity. NULL uses 0.5 when overlaid groups overlap (bar_mode = "overlay" with more than one nonempty group outside ridge mode) and 0.75 otherwise.
border_alpha	Numeric [0, 1]: Opacity of the bar borders, drawn in the series color. 0 draws no border.
na_rm	Logical: Whether to remove NA values before computing densities.
verbosity	Integer [0, Inf): Verbosity level for removed-NA messages.
mode	Character {"overlap", "ridge"}: Overlay samples or align rows on common x and y scales. Ridge rows use sample names in place of a legend. Increase figure height for many rows.
order	Character {"input", "mean", "median"}: Sample order. Summary orders are decreasing, retain ties in input order, and place empty samples last.
bin_stat	Character {"count", "sum", "mean", "min", "max"}: Statistic of x observations in each bin. Empty bins have height zero. Non-count statistics require normalization = "count" and density = FALSE.
bar_mode	Character {"overlay", "group", "stack"}: Histogram group layout. Stacks accumulate positive and negative values separately. Group/stack layouts require mode = "overlap" and density = FALSE.
zoom	Logical, DataZoom, or list of DataZoom: Enable x-axis zoom. TRUE adds a slider below the axis title, Shift+wheel zoom, and drag panning; in ridge mode, every row zooms together and no slider is drawn. Double-click resets the view. FALSE disables zoom. Pass DataZoom objects (or a list of them) for full control.

Details

Normalization is within each sample: "count" is the raw count; "probability" and "percent" divide by sample size (and multiply by 100 for percent). "density" divides by sample size and bin width, so total area is one. "count_density" divides only by width, so area is sample size. Unequal-width bins require one of the two density scales.

With density = TRUE, each curve uses the same settings as draw_density() and is scaled to the chosen histogram units. Count, probability, and percent curves multiply the estimated density by the common bin width and the appropriate sample-size factor. Density/count-density curves need no width factor. The histogram and curve share a color and legend toggle.

Value

htmlwidget: Widget object.

Examples

```
draw_histogram(iris[["Sepal.Length"]], xlab = "Sepal length")
```

---

draw_learning_curve *Draw a Learning Curve from Recorded Losses*

Description

Draw training and validation losses against training progress using a LineConfig. This function accepts an ordinary table and does not require rtemis. Supply a data frame containing its recorded learning curve.

Usage

```
draw_learning_curve(
  data,
  unit = attr(data, "unit", exact = TRUE),
  selected = attr(data, "selected", exact = TRUE),
  xlab = NULL,
  ylab = "Loss",
  title = NULL,
  theme = NULL,
  width = NULL,
  height = NULL,
  element_id = NULL,
  filename = NULL,
  ...,
  legend_position = "top",
  legend_placement = "outside",
  legend_title = NULL
)
```

Arguments

data	Data frame: Numeric iteration and at least one of loss_training or loss_validation. An optional tree column identifies ensemble members. Iterations must be unique within each tree.
unit	Optional Character: Unit of progress, such as "epochs" or "leaves". Read from the table's unit attribute when omitted. NULL uses "Iteration" for the axis label.
selected	Optional Numeric: Selected step. Read from the table's selected attribute when omitted. NA or NULL means no selected step.

xlab	Optional Character: Horizontal axis label, overriding unit.
ylab	Optional Character: Vertical axis label.
title	Optional Character: Chart title.
theme	Optional Theme: Theme override. Set palette in ... to override its series colors.
width	Optional Character or Numeric: Widget width.
height	Optional Character or Numeric: Widget height.
element_id	Optional Character: Explicit DOM element ID for the widget container. NULL lets htmlwidgets generate one.
filename	Optional Character: If provided, save the widget to this file via save_drawing().
...	Additional appearance and axis arguments to setup_LineConfig(), such as palette, zoom, points, xlim, and margin_top. The data bindings are determined by the loss table.
legend_position	Character {"top", "bottom", "left", "right", "top-left", "top-right", "bottom-left", "bottom-right"}: Legend anchor. Top/bottom anchors use horizontal rows; left/right anchors use a vertical column. Corner anchors align within the top or bottom row.
legend_placement	Character {"outside", "inside"}: Relation to the plotting area. Outside placement reserves space for the complete legend; inside placement overlays the data. Neither setting adds a missing legend.
legend_title	Optional Character: Title drawn with the legend or colorbar: above left/right legends and vertical colorbars, before the first row of top/bottom legends and horizontal colorbars. NULL draws no title. A title does not add a missing legend.

Details

Steps are sorted numerically. With a tree column, losses are averaged at each step over trees with a nonmissing loss for that series. Trees that have stopped contribute no value at later steps. Entirely missing series are omitted; missing values within a series remain gaps.

The selected point uses training loss if that series exists, otherwise validation loss. An absent step or missing loss produces no selected marker. selected = NULL disables it. The marker follows the line config's points setting, so points = FALSE hides it along with the other point symbols.

For another interface, materialize columns iteration, Training, Validation, and optionally Selected (missing except at the selected step), omitting unavailable loss columns. Bind these with setup_LineConfig() using x = "iteration" and y naming the loss columns. Store the unit as xlab. That data and config reproduce the chart without R attributes or a fitted model; aggregation is performed before rendering.

Value

htmlwidget: ECharts learning curve.

Examples

```
losses <- data.frame(
  iteration = 1:4,
  loss_training = c(4, 2, 1, 0.5),
  loss_validation = c(4.5, 2.5, 2, 2.2)
)
draw_learning_curve(losses, unit = "epochs", selected = 3)
```

---

draw_line *Draw a Line Chart*

Description

Quick line chart from x/y data.

Usage

```
draw_line(
  x,
  y,
  names = NULL,
  group = NULL,
  smooth = FALSE,
  area = FALSE,
  points = TRUE,
  blocks = NULL,
  block_color = NULL,
  block_opacity = 0.2,
  palette = NULL,
  line_style = NULL,
  pad = DEFAULT_PAD,
  xlim = NULL,
  ylim = NULL,
  square = FALSE,
  equal_axes = FALSE,
  xlab = NULL,
  ylab = NULL,
  title = NULL,
  theme = NULL,
  zoom = FALSE,
  margins = DEFAULT_MARGINS,
  width = NULL,
  height = NULL,
  element_id = NULL,
  filename = NULL,
  legend_position = "top",
  legend_placement = "outside",
```

```
  legend_title = NULL
)
```

Arguments

x	Vector: X-axis values. Numeric values get a value axis. Date and POSIXct values get a time axis: points are spaced by elapsed time and the labels are chosen adaptively for the span shown (years, then months, then days, down to seconds), and they read as the timestamps do in R whatever timezone the chart is viewed in. Anything else gets a category axis with one slot per distinct value, in order of appearance.
y	Numeric or named list: Y values.
names	Optional Character: Series names used when y is an unnamed list.
group	Optional Atomic vector or single-column data frame: Grouping variable, one value per point. Splits a single y vector into one line per level, named by the level, as draw_scatter() does for points. Cannot be combined with a list y or with blocks. Points whose group is NA are dropped.
smooth	Logical: Whether to smooth lines.
area	Logical: Whether to show area fill.
points	Logical: Whether to show point markers on each data value. Defaults to TRUE; set to FALSE on long time-series where the symbols add visual noise.
blocks	Optional factor, integer, character, or logical of length length(x): Per-x-value group label used to shade vertical background bands. Contiguous runs of the same level become one band. NA entries produce no band.
block_color	Optional Character vector or list of length k, where k is the number of unique levels in blocks: Fill color for each level. Match by name to factor levels if named, else positional. NA, NULL, or "transparent" entries skip that level.
block_opacity	Numeric [0, 1]: Fill opacity applied to all bands. Defaults to 0.2.
palette	Optional Character: Series color palette — a single color string or character vector that overrides the theme palette for this chart. color takes precedence over the theme palette (it sets option.color).
line_style	Optional Character {"solid", "dashed", "dotted"}: Line dash style — one value per series, recycled to match the number of series.
pad	Numeric [0, Inf): Fraction of the data range to extend each axis by when xlim / ylim are not given. The default matches base R's xaxs = "r", which extends the range by 4% at each end. Defaults to the range of all y values across series (no padding).
xlim	Optional Numeric, Date, or POSIXct [length 2]: X-axis limits c(min, max). Only supported when x is numeric or a time; passing xlim with a categorical x errors. On a time axis, give it in the same class as x (or as epoch milliseconds). Defaults to the range of x extended by pad.
ylim	Optional Numeric [length 2]: Y-axis limits c(min, max).
square	Logical: If TRUE, draw the plotting box square – equal height and width in pixels, excluding axis labels and margins.

`equal_axes` Logical: If TRUE, give one data unit the same size in pixels on both axes. Set both for a plot that is square *and* to scale, such as an ROC curve; the two axes are then made to span the same interval. Requires a numeric `x`. See Details.

`xlab` Optional Character: X-axis title.

`ylab` Optional Character: Y-axis title.

`title` Optional Character: Chart title.

`theme` Optional Theme: Theme override. The palette inside the theme can be overridden per-chart with the `color` argument.

`zoom` Logical, DataZoom, or list of DataZoom: Enable x-axis zoom. `TRUE` adds a slider below the axis title, Shift+wheel zoom, and drag panning on the x-axis; double-click resets the view. `FALSE` disables zoom. Pass DataZoom objects (or a list of them) for full control over zoom behavior, styling, and placement.

`margins` Optional Named numeric vector or named list: Plot margins in pixels (or percentage strings) for any of `"top"`, `"right"`, `"bottom"`, `"left"` — e.g. `c(left = 80, right = 20)` or `list(left = 80, right = "10%")`. Unspecified sides keep echarts' default auto-sizing (`outerBoundsMode = "same"`), which shrinks the grid to fit axis labels and names with no dead space.

`width` Optional Character or Numeric: Widget width.

`height` Optional Character or Numeric: Widget height.

`element_id` Optional Character: Explicit DOM element ID for the widget container. `NULL` lets htmlwidgets generate one.

`filename` Optional Character: If provided, save the widget to this file via `save_drawing()`.

`legend_position` Character {"top", "bottom", "left", "right", "top-left", "top-right", "bottom-left", "bottom-right"}: Legend anchor. Top/bottom anchors use horizontal rows; left/right anchors use a vertical column. Corner anchors align within the top or bottom row.

`legend_placement` Character {"outside", "inside"}: Relation to the plotting area. Outside placement reserves space for the complete legend; inside placement overlays the data. Neither setting adds a missing legend.

`legend_title` Optional Character: Title drawn with the legend or colorbar: above left/right legends and vertical colorbars, before the first row of top/bottom legends and horizontal colorbars. `NULL` draws no title. A title does not add a missing legend.

Details

Axis baselines emphasize zero only when it is in the visible orthogonal range. Categorical x axes follow the same rule as numeric and time axes.

Value

htmlwidget: Widget object.

Square and equally-scaled plots

`square` and `equal_axes` are separate requests. `square` is about the *shape* of the plotting box: equal height and width in pixels, measured on the box itself, not including axis labels or margins. `equal_axes` is about *scale*: one data unit occupies the same number of pixels on both axes, so a 45-degree line really is a slope of 1.

Either works alone. `equal_axes` on a chart whose x spans ten units and whose y spans one gives a wide, short box, which is what equal scaling means there.

Asking for both is a statement about the limits, since the only square box in which both axes are equally scaled is one whose axes span the same interval. So both axes are made to span one interval: over all the values when neither limit was given, or the one you gave when you gave exactly one. Giving both `xlim` and `ylim` as different intervals is an error rather than a silent override.

The box itself is solved in the browser, which is the only side that knows how wide the container is, and re-solved whenever it resizes.

Examples

```
# An ROC curve is square by nature, on axes that share their [0, 1] range.
draw_line(
  x = c(0, 0.2, 0.6, 1),
  y = c(0, 0.55, 0.85, 1),
  square = TRUE,
  equal_axes = TRUE,
  xlab = "False positive rate",
  ylab = "True positive rate"
)
```

`draw_manhattan` *Draw a Categorical Manhattan Plot*

Description

Plot signed estimates and raw p-values through SignificanceConfig. The Manhattan view places outcomes in input order; it does not use genomic coordinates. Missing pairs have no marks but remain in the testing family.

Usage

```
draw_manhattan(
  x,
  pvals,
  xnames = NULL,
  p_adjust_method = "holm",
  p_transform = "neg_log10",
  p_thresh = 0.05,
  x_thresh = 0,
```

```
  annotate_n = 7L,
  ...,
  legend_position = "top",
  legend_placement = "outside",
  legend_title = NULL,
  theme = NULL,
  width = NULL,
  height = NULL,
  element_id = NULL,
  filename = NULL,
  group = NULL,
  zoom = FALSE
)
```

Arguments

x	Numeric vector: Signed estimates; NA values have no plotted mark.
pvals	Numeric vector: Raw p-values in [0, 1], or NA.
xnames	Optional Character vector: One outcome label per estimate.
p_adjust_method	Character: Method from stats::p.adjust.methods. Adjustment precedes transformation.
p_transform	Character {"neg_log10", "identity", "one_minus"}: Named display transform of adjusted p-values.
p_thresh	Numeric (0, 1]: Significance threshold on adjusted p-values; the comparison is strict.
x_thresh	Numeric: Effect threshold separating low and high significant results.
annotate_n	Integer [0, Inf): Maximum significant feature labels per side in the volcano view, ranked by raw p-value with input-order ties.
...	Additional named settings for setup_SignificanceConfig(), such as reference, zero_cap, n_tests, colors, axis limits, or margins.
legend_position	Character {"top", "bottom", "left", "right", "top-left", "top-right", "bottom-left", "bottom-right"}: Legend anchor. Top/bottom anchors use horizontal rows; left/right anchors use a vertical column. Corner anchors align within the top or bottom row.
legend_placement	Character {"outside", "inside"}: Relation to the plotting area. Outside placement reserves space for the complete legend; inside placement overlays the data. Neither setting adds a missing legend.
legend_title	Optional Character: Title drawn with the legend or colorbar: above left/right legends and vertical colorbars, before the first row of top/bottom legends and horizontal colorbars. NULL draws no title. A title does not add a missing legend.
theme	Optional Theme: Widget/export theme.
width, height	Optional Numeric or Character: Widget dimensions.

element_id	Optional Character: HTML element identifier.
filename	Optional Character: Static output path, currently SVG.
group	Optional vector: One display-group label per outcome; unset uses significance and direction. Missing groups omit marks but retain the testing family.
zoom	Logical: Enable Shift+wheel zoom and drag panning; see setup_SignificanceConfig().

Value

An ECharts htmlwidget.

Statistical semantics

P-values are adjusted once over the full supplied family before rows with missing estimates or p-values are omitted from the marks. Missing hypotheses still count in n_tests; they do not become observed p-values of one. Raw and adjusted values remain in the compiled data. Function-valued transforms are not accepted by this portable configuration.

Under neg_log10, zero p-values have infinite mathematical height. They retain their zero value but are displayed at zero_cap, identified by triangles and a caption. No positive p-value is clamped. For identity and one-minus transforms, zeros are finite and no cap is used.

Examples

```
draw_manhattan(c(-2, 0.1, 3), c(0.001, 0.6, 0.002),
  xnames = c("A", "B", "C"), p_adjust_method = "holm")
```

---

draw_map *Render a MapModel as a MapLibre choropleth htmlwidget*

Description

Mid-level builder: takes a MapModel (or a plain list with rows, resolution, ...) plus styling, assembles a MapLibreOption, and dispatches through draw(). Most users call draw_choropleth() instead.

Usage

```
draw_map(
  model,
  classification = "quantile",
  colormap = "blues",
  num_classes = 5,
  opacity = 1,
  show_boundaries = TRUE,
  outline_width = 0.2,
  show_legend = TRUE,
  legend_position = "bottom-right",
```

```
  tooltip_position = "top-right",
  title = NULL,
  theme = NULL,
  width = NULL,
  height = NULL,
  element_id = NULL,
  filename = NULL
)
```

Arguments

model	MapModel or named list: The choropleth data to render.
classification	Character {"quantile", "equal", "jenks"}: Class-break method. "quantile" (equal counts), "equal" (equal intervals), or "jenks" (natural breaks).
colormap	Character: Color ramp. One of the sequential schemes "blues", "viridis", "ylorrd", "greens", "magma", or the diverging schemes "rdbu", "rdylgn", "spectral", "brbg".
num_classes	Numeric [2, 12]: Number of color classes.
opacity	Numeric [0, 1]: Region fill opacity.
show_boundaries	Logical: Whether to draw region outlines.
outline_width	Numeric [0, Inf): Region outline width in pixels.
show_legend	Logical: Whether to show the legend.
legend_position	Character {"top-left", "top-right", "bottom-left", "bottom-right"}: Legend corner.
tooltip_position	Character {"top-left", "top-right", "bottom-left", "bottom-right"}: Hover tooltip corner.
title	Optional Character: Title (currently surfaced via the value label; reserved for future use).
theme	Optional Theme, list, or NA: Theme override. NULL enables light/dark auto-detection (matching draw()).
width	Optional Character or Numeric: Widget width.
height	Optional Character or Numeric: Widget height.
element_id	Optional Character: Explicit element ID.
filename	Optional Character: Static export of map widgets is not yet supported. Supplying a path raises an error.

Value

htmlwidget.

Examples

```
model <- MapModel(
  rows = list(
    MapRow(location = "USA", value = 25.5),
    MapRow(location = "CAN", value = 12.1)
  )
)
draw_map(model)
```

`draw_mermaid` *Draw a Mermaid Diagram*

Description

Renders Mermaid diagram text with Pele (https://github.com/obsidianmd/pele) as a static SVG in an htmlwidget, shown in the RStudio and VS Code Viewer panes, Quarto and R Markdown documents, and Shiny. With `filename`, it also writes the diagram to an SVG file.

Usage

```
draw_mermaid(
  source,
  palette = NULL,
  node_color = NULL,
  fill_alpha = 0.1,
  font_size = 14,
  padding = NULL,
  auto_direction = FALSE,
  fit = FALSE,
  title = NULL,
  theme = NULL,
  width = NULL,
  height = NULL,
  element_id = NULL,
  filename = NULL
)
```

Arguments

`source`	Character: Mermaid diagram text, as one string or one element per line.
`palette`	Optional Character: Colors for categorical entities, such as pie slices and mindmap branches. `NULL` uses the theme palette.
`node_color`	Optional Named character: Colors for selected nodes, named by node id, e.g. `c(train = rtemis_colors[["teal"]])`. A node styled in the source with `style` or `classDef` keeps that style.
`fill_alpha`	Numeric [0, 1]: Fill opacity of the nodes in `node_color`.

font_size	Numeric (0, Inf): Label font size in pixels.
padding	Optional Numeric [0, Inf): Space around the diagram in pixels. NULL uses Pele's default.
auto_direction	Logical: Allow Pele to draw left-to-right flowcharts top-to-bottom in widgets narrower than 640 pixels.
fit	Logical: Shrink the diagram to the widget width, with pinch and Ctrl/Cmd-wheel zoom, instead of drawing it at its natural size.
title	Optional Character: Diagram title, written into the source's frontmatter. It replaces a title the source states.
theme	Optional Theme, list, or NA: Theme override. NULL enables light/dark auto-detection.
width	Optional Character or Numeric: Widget width.
height	Optional Character or Numeric: Widget height. NULL fills the Viewer pane and takes the diagram's height in documents.
element_id	Optional Character: Explicit element ID.
filename	Optional Character: Path of an SVG file to write with save_drawing(). The SVG has the diagram's natural size.

Details

The diagram renders at its natural size, so labels keep font_size however long or wide the diagram is, and the widget scrolls. In the Viewer pane the widget fills the pane. In a document with height unset, the widget takes the diagram's height and scrolls horizontally when the diagram is wider than the page. fit = TRUE shrinks the diagram to the widget width instead.

Uncolored elements use neutral grays that follow the light or dark theme. Categorical entities use palette, or the theme palette. node_color highlights selected nodes: each gets a border in its color and a fill of the same color at fill_alpha opacity, so its label keeps the theme's text color. It applies to the nodes of flowchart, state, class, entity relationship, block, architecture, and kanban diagrams, which Pele draws with the neutral fill and border.

Value

htmlwidget.

Supported Mermaid syntax

Pele supports every diagram type built into Mermaid 12.1.0, including flowchart, sequence, class, state, entity relationship, Gantt, git graph, pie, mindmap, kanban, timeline, user journey, quadrant chart, XY chart, requirement, Sankey, architecture, C4, block, packet, radar, treemap, and Venn diagrams. The syntax and the per-type differences are listed in Pele's compatibility notes, https://github.com/obsidianmd/pele/blob/1.6.0/website/src/content/docs/compatibility.md; the Mermaid documentation examples of every type, as Mermaid source, are in https://github.com/obsidianmd/pele/tree/1.6.0/tests/corpus.

Not supported: Mermaid themes and themeVariables, ZenUML and other external Mermaid plugins, HTML labels, and JavaScript callbacks in click statements. Configuration support varies by diagram type.

Pele computes its own layout, so node positions, spacing, and edge paths differ from Mermaid's. A flowchart subgraph keeps its own `direction` when it links to nodes outside the subgraph. Pele renders at most 50,000 source characters and 5,000 edges in a flowchart.

Pele uses at most eight categorical colors and repeats them from the ninth category. Colors set in the source with `style` or `classDef` override the theme and stay the same in light and dark mode.

Author(s)

EDG

Examples

```
# Train/test split
draw_mermaid(c(
  "flowchart TB",
  "  data[(Data)] --> split[Split]",
  "  split --> train[(Training set)]",
  "  split --> test[(Test set)]",
  "  train --> fit[Train model]",
  "  fit -->|trained model| eval[Evaluate on test set]",
  "  test --> eval"
), node_color = c(
  train = rtemis_colors[["teal"]],
  test = rtemis_colors[["magenta"]]
))

# Train/test split with hyperparameter tuning by resampling of the
# training set
draw_mermaid(c(
  "flowchart TB",
  "  data[(Data)] --> split[Split]",
  "  split --> train[(Training set)]",
  "  split --> test[(Test set)]",
  "  subgraph tuning[Resampling: tuning]",
  "    direction TB",
  "    inner_split[Split j = 1, ..., m] --> inner_train[(Inner training set j)]",
  "    inner_split --> val[(Validation set j)]",
  "    inner_train --> inner_fit[Train one model per hyperparameter set]",
  "    inner_fit --> inner_eval[Evaluate on validation set j]",
  "    val --> inner_eval",
  "  end",
  "  train --> inner_split",
  "  inner_eval -->|validation metrics| select[Select hyperparameters]",
  "  select -->|selected hyperparameters| fit[Train on training set]",
  "  train --> fit",
  "  fit -->|trained model| eval[Evaluate on test set]",
  "  test --> eval"
), node_color = c(
  train = rtemis_colors[["teal"]],
  inner_train = rtemis_colors[["teal"]],
  val = rtemis_colors[["purple"]],
  test = rtemis_colors[["magenta"]]
```

```
))

# Nested resampling: the inner loop tunes hyperparameters on each outer
# training set, and the outer loop estimates generalization performance
draw_mermaid(c(
  "flowchart TB",
  "  data[(Data)] --> split[Outer split i = 1, ..., k]",
  "  subgraph outer[Outer resampling: evaluation]",
  "    direction TB",
  "    split --> train[(Outer training set i)]",
  "    split --> test[(Outer test set i)]",
  "    subgraph tuning[Inner resampling: tuning]",
  "      direction TB",
  "      inner_split[Inner split j = 1, ..., m] --> inner_train[(Inner training set j)]",
  "      inner_split --> val[(Validation set j)]",
  "      inner_train --> inner_fit[Train one model per hyperparameter set]",
  "      inner_fit --> inner_eval[Evaluate on validation set j]",
  "      val --> inner_eval",
  "    end",
  "    train --> inner_split",
  "    inner_eval -->|validation metrics| select[Select hyperparameters]",
  "    select -->|selected hyperparameters| fit[Train on outer training set i]",
  "    train --> fit",
  "    fit -->|trained model i| eval[Evaluate on outer test set i]",
  "    test --> eval",
  "  end",
  "  eval -->|test metrics, all i| summary[Mean and SD of test metrics]"
), node_color = c(
  train = rtemis_colors[["teal"]],
  inner_train = rtemis_colors[["teal"]],
  val = rtemis_colors[["purple"]],
  test = rtemis_colors[["magenta"]]
))

# Pie chart of a data split using a single string for the source
draw_mermaid(
  "pie\n  \"Train\" : 70\n  \"Validation\" : 15\n  \"Test\" : 15",
  title = "Data split",
  palette = c(
    rtemis_colors[["teal"]],
    rtemis_colors[["purple"]],
    rtemis_colors[["magenta"]]
  )
)
```

`draw_metric` *Draw a Metric's Distribution Across Resamples*

Description

Draw stored fold scores using BoxplotConfig, with every available observation overlaid. Fold scores carry equal weight; this is a distribution of recorded scores, not a metric recomputed from pooled predictions or a confidence interval. Negative and unbounded metrics such as R-squared are not clipped to the unit interval.

Usage

```
draw_metric(
  data,
  metric = NULL,
  boxpoints = "all",
  horizontal = FALSE,
  xlab = if (horizontal) labelify(metric) else NULL,
  ylab = if (!horizontal) labelify(metric) else NULL,
  theme = NULL,
  width = NULL,
  height = NULL,
  element_id = NULL,
  filename = NULL,
  ...,
  legend_position = "top",
  legend_placement = "outside",
  legend_title = NULL
)
```

Arguments

`data`	Data frame: Long records with character `fold`, `split`, and `metric` identities, and numeric `value` scores (finite or NA).
`metric`	Optional Character: Metric to select. May be omitted when the table contains one metric.
`boxpoints`	Character {"none", "all", "outliers"}: Values to overlay.
`horizontal`	Logical: Draw horizontal distributions.
`xlab, ylab`	Optional Character: Axis labels. The value axis derives its label from the selected metric when omitted; explicit NULL omits it.
`theme`	Optional Theme: Theme override. The palette inside the theme can be overridden per-chart with the `color` argument.
`width`	Optional Character or Numeric: Widget width.
`height`	Optional Character or Numeric: Widget height.
`element_id`	Optional Character: Explicit DOM element ID for the widget container. `NULL` lets htmlwidgets generate one.
`filename`	Optional Character: If provided, save the widget to this file via `save_drawing()`.
`...`	Additional arguments to `setup_BoxplotConfig()`, such as `quartiles`, `whisker`, `point_spread`, `palette`, and `title`.

`legend_position`
: Character {"top", "bottom", "left", "right", "top-left", "top-right", "bottom-left", "bottom-right"}: Legend anchor. Top/bottom anchors use horizontal rows; left/right anchors use a vertical column. Corner anchors align within the top or bottom row.

`legend_placement`
: Character {"outside", "inside"}: Relation to the plotting area. Outside placement reserves space for the complete legend; inside placement overlays the data. Neither setting adds a missing legend.

`legend_title`
: Optional Character: Title drawn with the legend or colorbar: above left/right legends and vertical colorbars, before the first row of top/bottom legends and horizontal colorbars. `NULL` draws no title. A title does not add a missing legend.

Details

Select one metric per chart. Sample and fold order follow first appearance. Absent sample/fold pairs and explicit NA scores remain missing; available values alone define each box. Entirely unavailable samples retain an empty category, and missing values are reported in the console. Entirely unavailable input is an error. See `draw_boxplot()` for exact quartile, whisker, and point-placement semantics.

The portable result uses one numeric column per split and one observation identifier column, bound through `setup_BoxplotConfig()`. No fitted object, R callback, or attribute is needed to render it in another interface.

Value

htmlwidget: Metric distributions with optional point overlays.

Examples

```
scores <- data.frame(fold = rep(c("Fold1", "Fold2", "Fold3"), 2),
  split = rep(c("Training", "Test"), each = 3), metric = "rsq",
  value = c(.9, .85, .8, .7, .65, .6))
draw_metric(scores)
```

---

draw_network *Draw a Network / Graph*

Description

Render a network with Sigma.js. Accepts either a square weight / adjacency matrix (e.g. a correlation matrix) or an edge-list data frame, dispatching on the type of `x`. Communities are detected with Louvain and laid out with ForceAtlas2 by default.

Usage

```
draw_network(
  x,
  nodes = NULL,
  directed = FALSE,
  threshold = NULL,
  self_loops = FALSE,
  layout = "force",
  node_size = 10,
  edge_scale = 3,
  node_opacity = 0.95,
  edge_opacity = 0.4,
  show_labels = TRUE,
  scale_by_degree = TRUE,
  color_by_group = FALSE,
  resolution = 1,
  blend_edges = FALSE,
  palette = palette_colors(rtemis_colors),
  node_color = rtemis_colors[["teal"]],
  positive_color = NULL,
  negative_color = NULL,
  title = NULL,
  theme = NULL,
  width = NULL,
  height = NULL,
  element_id = NULL,
  filename = NULL
)
```

Arguments

x	Numeric matrix or data.frame: A square weight / adjacency matrix, or an edge-list data frame with `source` / `target` columns (and optional `weight`, `sign`). For a matrix, dimnames supply node ids and `[i, j]` is the edge weight (zero / `NA` entries produce no edge); the absolute value drives thickness and the sign drives color. For an edge list, `source` and `target` name the endpoints (the first two columns if those names are absent), with optional `weight` and `sign` columns.
nodes	Optional Data frame: Node attributes (`id`/`name`, `label`, `value`, `group`). Used only with edge-list input; ignored for matrix input.
directed	Logical: Whether the graph is directed.
threshold	Optional Numeric [0, Inf): For matrix input, drop edges whose absolute weight is below this value.
self_loops	Logical: For matrix input, whether to keep diagonal self-edges.
layout	Character {"force", "circular", "circlepack", "random"}: Layout algorithm. `"force"` is ForceAtlas2.
node_size	Numeric [0, Inf): Base node radius in screen pixels.

edge_scale	Numeric [0, Inf): Multiplier mapping normalized edge weight to stroke width.
node_opacity	Numeric [0, 1]: Node fill opacity.
edge_opacity	Numeric [0, 1]: Edge stroke opacity.
show_labels	Logical: Whether to render node labels.
scale_by_degree	Logical: Scale each node's radius by its value (weighted degree); when FALSE all nodes use the base size.
color_by_group	Logical: Color nodes by detected community (Louvain) instead of a single hue.
resolution	Numeric [0, Inf): Louvain resolution; higher yields more, smaller communities.
blend_edges	Logical: Color each edge as the blend of its two endpoint node colors instead of by sign.
palette	Character: Categorical colors for communities.
node_color	Character: Single node color used when color_by_group is FALSE.
positive_color	Optional Character: Edge color for positive-sign edges. NULL uses the theme's positive color.
negative_color	Optional Character: Edge color for negative-sign edges. NULL uses the theme's negative color.
title	Optional Character: Title shown above the network.
theme	Optional Theme, list, or NA: Theme override. NULL enables light/dark auto-detection (matching draw()).
width	Optional Character or Numeric: Widget width.
height	Optional Character or Numeric: Widget height.
element_id	Optional Character: Explicit element ID.
filename	Optional Character: Static export of network widgets is not yet supported. Supplying a path raises an error.

Value

htmlwidget.

Examples

```
# Correlation matrix
draw_network(cor(mtcars), threshold = 0.5)

# Edge list
edges <- data.frame(
  source = c("a", "a", "b"),
  target = c("b", "c", "c"),
  weight = c(1, 2, 0.5)
)
draw_network(edges)
```

`draw_panels` *Compose Drawings into One Figure*

Description

Arrange independent ECharts widgets in a row-major grid. Each panel keeps its own legend, tooltips, zoom, axes, and custom layers. No linked zoom or legend selection is implied. SVG export includes all panels in one vector file and uses the same layout calculation as the browser.

Usage

```
draw_panels(
  plots,
  ncol = 1L,
  gap = 12,
  padding = 0,
  layout = NULL,
  width = 900,
  height = NULL,
  element_id = NULL,
  filename = NULL
)
```

Arguments

`plots`	List: One or more ECharts widgets from draw_*() functions.
`ncol, gap, padding`	See PanelLayout.
`layout`	Optional PanelLayout: Complete layout overriding the three convenience settings; do not supply both forms.
`width`	Optional Character or Numeric: Total figure width.
`height`	Optional Numeric: Total figure height. Unset allocates 360 pixels per row plus padding and gaps.
`element_id`	Optional Character: Figure DOM identifier.
`filename`	Optional Character: Export the complete figure with `save_drawing()`.

Details

Child widget widths and heights are replaced by equally sized layout cells. Fixed-aspect and square-cell heatmap grids fit within both cell dimensions. Dendrograms remain aligned with their heatmap. Give panels enough space for their labels and legends. Explicit child themes are kept; automatic themes follow the browser, and resolve to light for SVG export. Padding and gaps use the common child background when all panels agree; mixed-background figures remain transparent. This also applies to SVG export. Nested layouts, Sigma/MapLibre, and child JavaScript hooks are not supported by this initial composition API.

Value

htmlwidget: A figure containing independent chart panels.

Examples

```
draw_panels(list(draw_bar(c("A", "B"), c(2, 4)),
  draw_boxplot(list(A = 1:5, B = 3:7), boxpoints = "all")), ncol = 2)
```

---

draw_pie *Draw a Pie Chart*

Description

Quick pie chart from values and labels.

Usage

```
draw_pie(
  values,
  labels = NULL,
  radius = "75%",
  rose_type = NULL,
  palette = NULL,
  title = NULL,
  theme = NULL,
  width = NULL,
  height = NULL,
  element_id = NULL,
  filename = NULL,
  legend_position = "top",
  legend_placement = "outside",
  legend_title = NULL,
  label_format = "name",
  percent_digits = 1L
)
```

Arguments

values	Numeric or data frame: Slice values, or a two-column table containing labels followed by values. Values must be finite and nonnegative with a positive total.
labels	Optional Character: Slice labels; names(values) when omitted.
radius	Numeric or Character: Pie radius.
rose_type	Optional Character {"radius", "area"}: Nightingale chart type.
palette	Optional Character: Series color palette — a single color string or character vector that overrides the theme palette for this chart. `color` takes precedence over the theme palette (it sets `option.color`).

title	Optional Character: Chart title.
theme	Optional Theme: Theme override. The palette inside the theme can be overridden per-chart with the color argument.
width	Optional Character or Numeric: Widget width.
height	Optional Character or Numeric: Widget height.
element_id	Optional Character: Explicit DOM element ID for the widget container. NULL lets htmlwidgets generate one.
filename	Optional Character: If provided, save the widget to this file via save_drawing().
legend_position	Character {"top", "bottom", "left", "right", "top-left", "top-right", "bottom-left", "bottom-right"}: Legend anchor. Top/bottom anchors use horizontal rows; left/right anchors use a vertical column. Corner anchors align within the top or bottom row.
legend_placement	Character {"outside", "inside"}: Relation to the plotting area. Outside placement reserves space for the complete legend; inside placement overlays the data. Neither setting adds a missing legend.
legend_title	Optional Character: Title drawn with the legend or colorbar: above left/right legends and vertical colorbars, before the first row of top/bottom legends and horizontal colorbars. NULL draws no title. A title does not add a missing legend.
label_format	Character {"name", "value", "percent", "name_value", "name_percent"}: Slice label content.
percent_digits	Integer [0, 6]: Percentage precision.

Value

htmlwidget: Widget object.

Examples

```
draw_pie(
  values = c(1048, 735, 580),
  labels = c("Chrome", "Firefox", "Safari")
)
```

---

draw_protein *Draw an Annotated Protein from Sequence and Annotation Inputs*

Description

Adapts sequence strings, residue vectors, protein records, and local JSON to the shared A3 representation used by draw_a3(). Named legacy annotation vectors enumerate one-based residues. Region vectors preserve contiguous runs; isolated region residues are drawn as sites. Modern specifications from rtemis.a3 annotation_position/annotation_range/annotation_variant are accepted. The common meander layout, theme, legends, and SVG path match rtemislive. Fetch accession records explicitly with rtemis.a3 before plotting; a sequence string never initiates a network request.

Usage

```
draw_protein(
  x,
  site = list(),
  region = list(),
  ptm = list(),
  cleavage_site = list(),
  variant = list(),
  disease_variants = NULL,
  ...
)
```

Arguments

x	Character, list, or A3: Sequence, protein record, local JSON path, or A3.
site, region, ptm, cleavage_site	List: Named annotation vectors or modern specifications.
variant	List: Records with position plus optional variant metadata.
disease_variants	Optional Numeric: Disease-associated residue positions.
...	Additional display settings passed to draw_a3().

Value

htmlwidget: Annotated protein diagram.

Examples

```
draw_protein("MAEPRQEFEVMEDHAGTYGLGDRK", site = list(Active = c(5L, 17L)),
  region = list(Domain = 3:10), ptm = list(Phosphorylation = 2L))
```

---

draw_pvals *Draw One-Minus-P-Value Bars*

Description

A convenience view of SignificanceConfig. Adjustment uses the full supplied testing family, including missing slots; missing p-values are not plotted. The reference is at one minus the significance threshold.

Usage

```
draw_pvals(
  x,
  xnames = names(x),
  p_adjust_method = "none",
  pval_hline = 0.05,
  ...
)
```

Arguments

x	Numeric: P-values in [0, 1] or NA.
xnames	Optional Character: Feature labels, otherwise names of x.
p_adjust_method	Character: A method from stats::p.adjust.methods.
pval_hline	Numeric (0, 1): Significance threshold.
...	Additional arguments passed to draw_manhattan().

Value

htmlwidget: Bars showing one minus adjusted p-values.

Examples

```
draw_pvals(c(A = .01, B = .2, C = .04), p_adjust_method = "holm")
```

---

draw_roc *Draw Receiver Operating Characteristic Curves*

Description

Accepts reference labels and predicted probabilities, matching lists of those inputs, or a long table with fpr, tpr, and optional auc, class, split, fold, and omitted columns. Named input lists are matched by name.

Usage

```
draw_roc(
  true_labels,
  predicted_prob = NULL,
  positive = NULL,
  ...,
  legend_position = "top",
  legend_placement = "outside",
  legend_title = NULL,
  theme = NULL,
```

```
  width = NULL,
  height = NULL,
  element_id = NULL,
  filename = NULL
)
```

Arguments

`true_labels`	Factor, vector, list, or data frame: Reference labels or ROC records.
`predicted_prob`	Optional Numeric vector, matrix, or list: Class probabilities.
`positive`	Optional Character: Positive class for binary input.
`...`	Additional named settings for `setup_ROCConfig()`.
`legend_position`	Character {"top", "bottom", "left", "right", "top-left", "top-right", "bottom-left", "bottom-right"}: Legend anchor. Top/bottom anchors use horizontal rows; left/right anchors use a vertical column. Corner anchors align within the top or bottom row.
`legend_placement`	Character {"outside", "inside"}: Relation to the plotting area. Outside placement reserves space for the complete legend; inside placement overlays the data. Neither setting adds a missing legend.
`legend_title`	Optional Character: Title drawn with the legend or colorbar: above left/right legends and vertical colorbars, before the first row of top/bottom legends and horizontal colorbars. `NULL` draws no title. A title does not add a missing legend.
`theme`	Optional Theme: Chart theme.
`width, height`	Optional Numeric or Character: Widget dimensions.
`element_id`	Optional Character: HTML element identifier.
`filename`	Optional Character: Static output path, currently SVG.

Details

Binary input uses the second factor level (second level after factor conversion for nonfactors), unless `positive` names the desired class. An unnamed probability vector/column describes that class. Named columns identify their classes; multiclass matrices use names, or factor-level order when unnamed. Multiclass curves are one-versus-rest. Missing labels/scores are excluded separately for each curve and disclosed. Scores must be probabilities in `[0, 1]`; direction is fixed, and tied scores enter together. No observations are deduplicated, curves smoothed, or vertices downsampled.

Value

An ECharts htmlwidget.

Legend placement

The shared top/outside layout reserves a horizontal band above the plot. Choose an inside corner to overlay the legend on the data.

Statistical semantics

Vertices are ordered by FPR then TPR and joined without smoothing. Curves must be monotone and include (0, 0) and (1, 1). Supplied AUC is authoritative and may describe a higher-resolution curve. Without an AUC binding, the supplied vertices define trapezoidal AUC. An undefined curve is represented by missing FPR, TPR, and AUC and is reported in the console when omitted. Per-resample legends report the unweighted mean and sample SD of defined fold AUCs, with available/total curve counts. SD is NA for one defined fold. Pooled AUC is never inferred from per-fold AUCs. The legend uses the shared position and placement controls. Curve identity and AUC share a line; long labels wrap to the available width. All entries remain present in static SVG output, without a scrolling legend. Hover describes the selected curve, including its resample identifier. Numeric tooltip labels use `digits`; coordinates and AUCs retain their full precision in the compiled data.

Examples

```
draw_roc(factor(c("no", "yes", "no", "yes")), c(.1, .8, .5, .5))
draw_roc(data.frame(fpr = c(0, 0, 1), tpr = c(0, 1, 1)))
```

`draw_sankey` *Draw a Sankey Diagram*

Description

Sankey diagram from a data frame of directed links. Node names are derived automatically from the unique values in the `source` and `target` columns; no separate node list is required.

Usage

```
draw_sankey(
  links,
  orient = "horizontal",
  node_width = NULL,
  node_gap = NULL,
  node_align = NULL,
  title = NULL,
  palette = NULL,
  link_color = "source",
  link_opacity = 0.45,
  margins = NULL,
  label_font_size = 12,
  theme = NULL,
  width = NULL,
  height = NULL,
  element_id = NULL,
  filename = NULL
)
```

Arguments

links	data.frame: Directed links with columns source (Character), target (Character), and value (Numeric).
orient	Optional Character {"horizontal", "vertical"}: Flow direction.
node_width	Optional Numeric [0, Inf): Node rectangle width in pixels.
node_gap	Optional Numeric [0, Inf): Vertical gap between nodes in pixels.
node_align	Optional Character {"justify", "left", "right"}: Node alignment within each column.
title	Optional Character: Chart title.
palette	Optional Character: Node color palette — a single color string or a character vector that overrides the theme palette for this chart.
link_color	Character {"source", "target", "gradient"} or a color: How ribbons are colored. "source", the default, gives each ribbon the color of the node it leaves, which is what lets a reader follow one source across the diagram.
link_opacity	Numeric [0, 1]: Ribbon opacity. Below 1 so that crossing ribbons blend rather than hide one another.
margins	Optional named list or numeric: Space around the plotting box, as any of left, right, top, bottom in pixels or as a CSS width. Sides left unnamed are derived from the labels: a node's label is drawn to its right, so the last column needs room or its labels are clipped. Set them explicitly when preparing a static export, where the exact canvas is known.
label_font_size	Numeric: Node label font size in pixels. Also what the derived margins are estimated from.
theme	Optional Theme: Theme override.
width	Optional Character or Numeric: Widget width.
height	Optional Character or Numeric: Widget height.
element_id	Optional Character: Explicit DOM element ID for the widget container. NULL lets htmlwidgets generate one.
filename	Optional Character: If provided, save the widget to this file via save_drawing().

Value

htmlwidget: Widget object.

Examples

```
links <- data.frame(
  source = c("A", "A", "B", "C"),
  target = c("B", "C", "D", "D"),
  value  = c(8, 4, 6, 3)
)
draw_sankey(links)
```

draw_scatter *Draw a Scatter Plot*

Description

Quick scatter plot from x/y data with optional fitted line and confidence band.

Usage

```
draw_scatter(
  x,
  y,
  size = NULL,
  group = NULL,
  fit = NULL,
  fit_params = NULL,
  se = TRUE,
  fit_alpha = 0.25,
  n_fit = 200L,
  palette = NULL,
  xlim = NULL,
  ylim = NULL,
  pad = DEFAULT_PAD,
  square = FALSE,
  equal_axes = FALSE,
  xlab = NULL,
  ylab = NULL,
  title = NULL,
  theme = NULL,
  margins = DEFAULT_MARGINS,
  width = NULL,
  height = NULL,
  element_id = NULL,
  filename = NULL,
  se_times = 1.96,
  rsq = FALSE,
  diagonal = FALSE,
  diagonal_color = NULL,
  legend_position = "top",
  legend_placement = "outside",
  legend_title = NULL,
  fit_name = NULL,
  rug = FALSE,
  hover = NULL,
  point_alpha = NULL,
  zoom = FALSE
)
```

Arguments

x	Numeric: X values.
y	Numeric: Y values.
size	Optional Numeric: Symbol sizes.
group	Optional Atomic vector or single-column data frame: Grouping variable for multiple series.
fit	Optional Character: Model fitted to y on x and drawn as a line. "glm" uses stats::glm() and "gam" uses mgcv::gam(). Any other name, matched case-insensitively, is an rtemis supervised learning algorithm, such as "LINAD", "CART" or "LightGBM", trained with its default hyperparameters; this requires the rtemis package (>= 1.4.1). NULL disables fitting. The fitted line and standard-error band are computed per group when group is provided.
fit_params	Optional Named list: Arguments passed to the learner named in fit, as argument-value pairs: to stats::glm() for "glm", e.g. list(family = "poisson"); to mgcv::gam() for "gam", where k sets the basis dimension of the smooth, e.g. list(k = 5); and to the rtemis setup function otherwise, e.g. list(max_leaves = 8) for "LINAD", whose arguments are listed in ?rtemis::setup_LINAD. NULL uses the learner's defaults.
se	Logical: Whether to show the standard-error band. Algorithms without standard errors (every rtemis algorithm except GLM, GAM and a few others) draw the line alone.
fit_alpha	Numeric [0, 1]: Opacity for the confidence-band fill.
n_fit	Integer [2, Inf): Number of evaluation points for the fit.
palette	Optional Character: Series color palette — a single color string or character vector that overrides the theme palette for this chart. When group is set, colors are assigned per group in order. color takes precedence over the theme palette.
xlim	Optional Numeric [length 2]: X-axis limits c(min, max). Defaults to range(x) extended by pad.
ylim	Optional Numeric [length 2]: Y-axis limits c(min, max). Defaults to range(y) extended by pad.
pad	Numeric [0, Inf): Fraction of the data range to extend each axis by when xlim / ylim are not given. The default matches base R's xaxs = "r", which extends the range by 4% at each end.
square	Logical: If TRUE, draw the plotting box square – equal height and width in pixels, excluding axis labels and margins.
equal_axes	Logical: If TRUE, give one data unit the same size in pixels on both axes. Set both for a plot that is square *and* to scale, such as a true-versus-predicted plot whose identity line should run at 45 degrees; the two axes are then made to span the same interval. See Details.
xlab	Optional Character: X-axis title.
ylab	Optional Character: Y-axis title.
title	Optional Character: Chart title.

theme	Optional Theme: Theme override. The palette inside the theme can be overridden per-chart with the color argument.
margins	Optional Named numeric vector or named list: Plot margins in pixels (or percentage strings) for any of "top", "right", "bottom", "left". Unspecified sides keep echarts' default auto-sizing. See draw_line() for details.
width	Optional Character or Numeric: Widget width.
height	Optional Character or Numeric: Widget height.
element_id	Optional Character: Explicit DOM element ID for the widget container. NULL lets htmlwidgets generate one.
filename	Optional Character: If provided, save the widget to this file via save_drawing().
se_times	Numeric [0, Inf): Standard-error multiplier for the band.
rsq	Logical: Include the fitted model's R-squared in series labels. This describes the overlay fit, not predictive performance against the identity line. Constant responses have undefined R-squared, labeled NA.
diagonal	Logical: Draw a dashed identity line within the axis limits.
diagonal_color	Optional Character: Identity-line color.
legend_position	Character {"top", "bottom", "left", "right", "top-left", "top-right", "bottom-left", "bottom-right"}: Legend anchor. Top/bottom anchors use horizontal rows; left/right anchors use a vertical column. Corner anchors align within the top or bottom row.
legend_placement	Character {"outside", "inside"}: Relation to the plotting area. Outside placement reserves space for the complete legend; inside placement overlays the data. Neither setting adds a missing legend.
legend_title	Optional Character: Title drawn with the legend or colorbar: above left/right legends and vertical colorbars, before the first row of top/bottom legends and horizontal colorbars. NULL draws no title. A title does not add a missing legend.
fit_name	Optional Character: Label for fitted layers.
rug	Logical: Draw marginal marks on both axes.
hover	Optional Character: One literal tooltip label per observation.
point_alpha	Optional Numeric [0, 1]: Point opacity. NULL sets it from the number of points drawn, as a rough guard against overplotting: 0.75 at 10 points, 0.57 at 150, 0.45 at 1,000 and 0.3 at 10,000, clamped to [0.15, 0.9]. Set a value to override.
zoom	Logical, DataZoom, or list of DataZoom: Enable zoom on both axes. TRUE adds Shift+wheel zoom and drag panning on both axes and a top-right toolbox with box-zoom, undo, and reset controls; double-click resets the view. Wheel zoom and panning scale both axes by the same factor; with square or equal_axes, the toolbox omits box zoom, which sets each axis independently. FALSE disables zoom. Pass DataZoom objects (or a list of them) for full control.

Value

htmlwidget: Widget object.

Square and equally-scaled plots

`square` and `equal_axes` are separate requests. `square` is about the *shape* of the plotting box: equal height and width in pixels, measured on the box itself, not including axis labels or margins. `equal_axes` is about *scale*: one data unit occupies the same number of pixels on both axes, so a 45-degree line really is a slope of 1.

Either works alone. `equal_axes` on a chart whose x spans ten units and whose y spans one gives a wide, short box, which is what equal scaling means there.

Asking for both is a statement about the limits, since the only square box in which both axes are equally scaled is one whose axes span the same interval. So both axes are made to span one interval: over all the values when neither limit was given, or the one you gave when you gave exactly one. Giving both `xlim` and `ylim` as different intervals is an error rather than a silent override.

The box itself is solved in the browser, which is the only side that knows how wide the container is, and re-solved whenever it resizes.

Examples

```
# True versus predicted: square box, equal scale, identity line at 45 deg.
draw_scatter(
  x = c(1, 2, 3, 4),
  y = c(1.1, 1.9, 3.4, 3.7),
  square = TRUE,
  equal_axes = TRUE,
  xlab = "True",
  ylab = "Predicted"
)
```

---

draw_scatter3d *Draw Interactive Three-Dimensional Points and Paths*

Description

Uses ECharts-GL in the browser and vector circles, paths, and text for SVG. The saved view uses the configured camera, not later browser rotations. Points omit missing coordinate/group rows together. Missing coordinates break paths within each group; groups keep their input order. A data frame or matrix can supply the coordinates in its first three columns, which must be numeric.

Usage

```
draw_scatter3d(
  x,
  y = NULL,
  z = NULL,
  group = NULL,
  ...,
  theme = NULL,
```

```
  width = NULL,
  height = NULL,
  element_id = NULL,
  filename = NULL
)
```

Arguments

x	Numeric, matrix, or data frame: X values, or three coordinate columns.
y, z	Optional Numeric: Y and Z values for vector x.
group	Optional vector: Group identities.
...	Additional settings passed to setup_Scatter3DConfig().
theme	Optional Theme: Theme override. The palette inside the theme can be overridden per-chart with the color argument.
width	Optional Character or Numeric: Widget width.
height	Optional Character or Numeric: Widget height.
element_id	Optional Character: Explicit DOM element ID for the widget container. NULL lets htmlwidgets generate one.
filename	Optional Character: If provided, save the widget to this file via save_drawing().

Value

htmlwidget: Interactive, exportable 3D drawing.

Examples

```
draw_scatter3d(iris[1:3], group = iris$Species)
```

---

draw_spectrogram *Draw a Spectrogram*

Description

Renders an interactive time-frequency spectrogram as an ECharts heatmap widget. Accepts either a raw signal vector (STFT is computed internally via signal::specgram()) or a pre-computed spectrogram matrix (freq x time).

Usage

```
draw_spectrogram(
  x,
  sample_rate = NULL,
  time = NULL,
  frequency = NULL,
  n_fft = 256L,
```

```
  window = "hanning",
  overlap = NULL,
  power = TRUE,
  db = TRUE,
  db_range = 80,
  freq_scale = "linear",
  freq_range = NULL,
  freq_unit = "Hz",
  time_range = NULL,
  time_unit = "s",
  colormap = "magma",
  colormap_reverse = FALSE,
  n_colors = 256L,
  zlim = NULL,
  show_colorbar = TRUE,
  colorbar_title = NULL,
  title = NULL,
  xlab = NULL,
  ylab = NULL,
  theme = NULL,
  margins = NULL,
  width = NULL,
  height = NULL,
  element_id = NULL,
  filename = NULL,
  verbosity = 1L,
  legend_position = "right",
  legend_placement = "outside",
  legend_title = NULL
)
```

Arguments

x	Numeric matrix (freq x time) or numeric vector (raw signal). A matrix is used directly; time and frequency vectors supply axis values (defaults to sample indices when absent). A complex matrix (raw STFT output, e.g. from signal::specgram()$S) is also accepted. A vector triggers STFT computation via signal::specgram(); sample_rate is required.
sample_rate	Optional Numeric (0, Inf): Sampling frequency in Hz. Required when x is a raw signal vector.
time	Optional Numeric: Time axis values in seconds, length ncol(x). Only used when x is a matrix.
frequency	Optional Numeric: Frequency axis values in Hz, length nrow(x). Only used when x is a matrix.
n_fft	Integer [2, Inf): FFT window size in samples. Passed to signal::specgram() as n. Only used when x is a raw signal.
window	Character {"hanning", "hamming", "blackman", "bartlett", "rectangular"} or

	Numeric: Window function name or a pre-built window vector. Passed to signal::specgram(). Only used when x is a raw signal.
overlap	Optional Integer [0, n_fft): Overlap between consecutive frames in samples. Defaults to n_fft / 2. Only used when x is a raw signal.
power	Logical: Treat spectral values as power (TRUE) or amplitude (FALSE). When x is complex, controls whether the STFT magnitude is squared (|S|^2) or left as-is (|S|). When db = TRUE, also determines the dB scaling for real matrices: 10 * log10() for power, 20 * log10() for amplitude.
db	Logical: Convert to dB. For power: 10 * log10(); for amplitude: 20 * log10(). Set FALSE when passing a pre-computed dB matrix.
db_range	Numeric (0, Inf): Dynamic range to display in dB below the spectral peak. Values below peak - db_range are clipped to the floor.
freq_scale	Character {"linear", "log"}: Frequency axis scale. With "log" the DC component (0 Hz) is automatically dropped.
freq_range	Optional Numeric[2]: Frequency range to display in Hz, e.g. c(20, 8000). Applied after STFT computation.
freq_unit	Character {"Hz", "kHz"}: Unit for the frequency axis.
time_range	Optional Numeric[2]: Time range to display in seconds.
time_unit	Character {"s", "ms"}: Unit for the time axis.
colormap	Character: Color palette. Accepts a **viridisLite** colormap name ("magma" (default), "inferno", "plasma", "viridis", "cividis", "mako", "rocket", "turbo"), "diverging" for a negative-background-positive scale in the theme's sign colors (suitable for signed data such as EEG/MEG amplitudes), or a character vector of >= 2 hex colors for a custom ramp.
colormap_reverse	
	Logical: Reverse the colormap direction.
n_colors	Integer [2, Inf): Number of discrete colors in the generated palette.
zlim	Optional Numeric[2]: Color-scale limits after all transformations (dB clipping, unit conversion). Defaults to the data range.
show_colorbar	Logical: Show the continuous visual-map colorbar.
colorbar_title	Optional Character: Colorbar label. Default: "dB", "Power", or "Amplitude" derived from db and power.
title	Optional Character: Chart title.
xlab	Optional Character: X-axis label. Default: "Time (s)" or "Time (ms)" depending on time_unit.
ylab	Optional Character: Y-axis label. Default: "Frequency (Hz)" or "Frequency (kHz)" depending on freq_unit.
theme	Optional Theme, list, or NA: Theme override passed to draw().
margins	Optional Named numeric or character vector / list: Plot margins in pixels. Valid names: "top", "right", "bottom", "left".
width	Optional Numeric or Character: Widget width.
height	Optional Numeric or Character: Widget height.

element_id Optional Character: Explicit DOM element ID for the widget container. NULL lets htmlwidgets generate one.

filename Optional Character: If provided, the widget is saved via save_drawing().

verbosity Integer [0, Inf): Verbosity level. 0 silences the large-spectrogram performance advisory.

legend_position
Character {"top", "bottom", "left", "right", "top-left", "top-right", "bottom-left", "bottom-right"}: Legend anchor. Top/bottom anchors use horizontal rows; left/right anchors use a vertical column. Corner anchors align within the top or bottom row.

legend_placement
Character {"outside", "inside"}: Relation to the plotting area. Outside placement reserves space for the complete legend; inside placement overlays the data. Neither setting adds a missing legend.

legend_title Optional Character: Title drawn with the legend or colorbar: above left/right legends and vertical colorbars, before the first row of top/bottom legends and horizontal colorbars. NULL draws no title. A title does not add a missing legend.

Details

Corresponds to HeatmapSeriesOption in src/chart/heatmap/HeatmapSeries.ts. ECharts docs: https://echarts.apache.org/en/option.html#series-heatmap

Value

htmlwidget

Examples

```
if (requireNamespace("signal", quietly = TRUE)) {
  t_vec <- seq(0, 2, by = 1 / 8000)
  sig   <- signal::chirp(t_vec, 200, 2, 2000)
  draw_spectrogram(sig, sample_rate = 8000)
}
```

---

draw_survfit *Draw a Fitted Survival Curve*

Description

Adapt single-state survival::survfit estimates to draw_survival(). Stratum labels, estimator-defined starting time, confidence bounds, and censor counts are preserved. Matrix-valued predicted curves must be selected one at a time; multistate and interval-censored fits require a different plotting contract.

Usage

```
draw_survfit(x, ..., risk_times = NULL, risk_table = !is.null(risk_times))
```

Arguments

x	survfit: Fitted right-censored or counting-process survival curves.
...	Additional named arguments for draw_survival().
risk_times	Optional Numeric vector: Times for risk-table counts. Automatic counts require an ordinary right-censored fit. For delayed entry, pass separately computed risk records through draw_survival().
risk_table	Logical: Draw risk counts. If requested without explicit times, uses approximately five evenly spaced times within the fit's range.

Value

An ECharts htmlwidget with native SVG export.

Examples

```
fit <- survival::survfit(survival::Surv(time, status) ~ sex, data = survival::lung)
draw_survfit(fit, risk_times = c(0, 250, 500, 750, 1000))
```

---

draw_survival *Draw Precomputed Survival Curves*

Description

Use draw_survfit() to draw fitted R survival objects. This interface accepts portable records and does not require the survival package.

Usage

```
draw_survival(
  data,
  ...,
  group = NULL,
  lower = NULL,
  upper = NULL,
  n_censor = NULL,
  n_risk = NULL,
  legend_position = "top",
  legend_placement = "outside",
  legend_title = NULL,
  theme = NULL,
  width = NULL,
  height = NULL,
  element_id = NULL,
  filename = NULL
)
```

Arguments

data	Data frame or list: Curve records, or curves and risk frames.
...	Additional named settings for setup_SurvivalConfig().
group, lower, upper, n_censor, n_risk	Optional Character: Column bindings. When omitted, a column with the matching name is used if present. Explicit NULL disables that binding.
legend_position	Character {"top", "bottom", "left", "right", "top-left", "top-right", "bottom-left", "bottom-right"}: Legend anchor. Top/bottom anchors use horizontal rows; left/right anchors use a vertical column. Corner anchors align within the top or bottom row.
legend_placement	Character {"outside", "inside"}: Relation to the plotting area. Outside placement reserves space for the complete legend; inside placement overlays the data. Neither setting adds a missing legend.
legend_title	Optional Character: Title drawn with the legend or colorbar: above left/right legends and vertical colorbars, before the first row of top/bottom legends and horizontal colorbars. NULL draws no title. A title does not add a missing legend.
theme	Optional Theme: Chart theme.
width, height	Optional Numeric or Character: Widget dimensions.
element_id	Optional Character: HTML element identifier.
filename	Optional Character: Static output path, currently SVG.

Value

An ECharts htmlwidget with native SVG export.

Records and statistical semantics

Supply a curve data frame, or a list with curves and optional risk data frames. Each curve row contains time and survival probability, with optional group, pointwise lower/upper confidence bounds, censor counts, and risk counts. Include the initial time/probability explicitly: the renderer never invents an origin. Records are sorted within groups; times must be unique and finite, probabilities nonincreasing in [0, 1]. Curves are right-continuous and stop at their last recorded time. Missing confidence bounds produce gaps, not zeros. A censor tick represents one or more censored observations at that time; its tooltip retains the supplied count. Counts may be fractional for weights. Median survival is the first crossing of 0.5; a plateau at 0.5 uses its midpoint, including the last follow-up time if the plateau is terminal. Comparisons to 0.5 use tolerance sqrt(.Machine$double.eps), as in survival. Landmarks outside a group's observed time range are omitted without extrapolation. Display precision never rounds plotted coordinates.

Risk records have fixed columns time, group, and n_risk, and state the risk count immediately before the specified time. They must cover the same unique times in each group. Unknown counts use NA and display as NA. The renderer does not interpolate or infer risk sets from the curves. All layers belonging to a group share its legend toggle, including risk rows.

Examples

```
draw_survival(data.frame(time = c(0, 1, 3), survival = c(1, .8, .4)))
```

`draw_ts` *Draw Time Series with Rolling Summaries*

Description

Draw Time Series with Rolling Summaries

Usage

```
draw_ts(
  x,
  time,
  window = 7L,
  group = NULL,
  roll_fn = "mean",
  align = "center",
  ...,
  theme = NULL,
  width = NULL,
  height = NULL,
  element_id = NULL,
  filename = NULL
)
```

Arguments

`x`	Numeric or named list: Observations.
`time`	Numeric, Date, POSIXct, or list: Shared or per-series times.
`window`	Integer `[1, Inf)`: Number of observations per rolling window.
`group`	Optional vector: Group identities for one series.
`roll_fn`	Character `{"none", "mean", "median", "min", "max", "sum"}`: Summary.
`align`	Character `{"center", "left", "right"}`: Window alignment.
`...`	Additional settings passed to `setup_TimeSeriesConfig()`.
`theme`	Optional Theme: Theme override. The palette inside the theme can be overridden per-chart with the `color` argument.
`width`	Optional Character or Numeric: Widget width.
`height`	Optional Character or Numeric: Widget height.
`element_id`	Optional Character: Explicit DOM element ID for the widget container. `NULL` lets htmlwidgets generate one.
`filename`	Optional Character: If provided, save the widget to this file via `save_drawing()`.

Value

htmlwidget: Raw observations and rolling curves with shared group legends.

Examples

```
draw_ts(BOD$demand, BOD$Time, window = 3L)
```

`draw_varimp` *Draw Variable Importance*

Description

Draw signed importance summaries with BarConfig, or fold distributions with BoxplotConfig using `type = "boxplot"`. Numeric input uses its names as variable labels, or row numbers when unnamed. Tables preserve named measures and optionally identify resamples with `fold`.

Usage

```
draw_varimp(
  x,
  measure = NULL,
  top_n = 20L,
  rank_by = "magnitude",
  summary = "mean",
  folds = NULL,
  horizontal = TRUE,
  xlab = NULL,
  ylab = NULL,
  title = NULL,
  theme = NULL,
  width = NULL,
  height = NULL,
  element_id = NULL,
  filename = NULL,
  type = "bar",
  decreasing = TRUE,
  ...,
  legend_position = "top",
  legend_placement = "outside",
  legend_title = NULL
)
```

Arguments

`x`	Numeric vector, single-column matrix, or data frame: Importance scores. Tables require `variable` and named numeric measures, optionally `fold`. Variable names must be unique within each fold.

measure	Optional Character: Measure column. NULL selects the first column other than variable or fold; vectors use "importance".
top_n	Optional Integer [1, Inf): Maximum number of variables. NULL includes all variables with a summary; fractions are not accepted.
rank_by	Character {"magnitude", "signed"}: Rank by absolute or signed summary, respectively.
summary	Character {"mean", "median"}: Summary across folds.
folds	Optional Character: Full set of fold IDs, including folds with no rows. NULL uses the IDs observed in the fold column.
horizontal	Logical: Draw horizontal bars or boxes.
xlab, ylab	Optional Character: Physical axis labels. NULL derives the score label from the measure and summary and labels the variable axis.
title	Optional Character: Chart title.
theme	Optional Theme: Theme override. Set palette in ... to override its series colors.
width	Optional Character or Numeric: Widget width.
height	Optional Character or Numeric: Widget height. NULL allocates space per selected variable for horizontal bars.
element_id	Optional Character: Explicit DOM element ID for the widget container. NULL lets htmlwidgets generate one.
filename	Optional Character: If provided, save the widget to this file via save_drawing().
type	Character {"bar", "boxplot"}: Summary or fold-distribution view.
decreasing	Logical: Select and display ranks from largest to smallest. FALSE selects and displays the smallest ranks first.
...	Additional settings for setup_BarConfig() or setup_BoxplotConfig(), according to type. Boxplots accept boxpoints, quartiles, whisker, and point styling; unset boxpoints shows all scores. Data bindings, labels, and orientation are set here.
legend_position	Character {"top", "bottom", "left", "right", "top-left", "top-right", "bottom-left", "bottom-right"}: Legend anchor. Top/bottom anchors use horizontal rows; left/right anchors use a vertical column. Corner anchors align within the top or bottom row.
legend_placement	Character {"outside", "inside"}: Relation to the plotting area. Outside placement reserves space for the complete legend; inside placement overlays the data. Neither setting adds a missing legend.
legend_title	Optional Character: Title drawn with the legend or colorbar: above left/right legends and vertical colorbars, before the first row of top/bottom legends and horizontal colorbars. NULL draws no title. A title does not add a missing legend.

Details

Resampled scores are summarized before selection. Magnitude ranking uses the absolute summary, not the mean absolute fold score. Signed ranking uses the summary itself. `decreasing = FALSE` selects the smallest ranks first. Ties retain first appearance in the input. The first selected variable appears at the top of horizontal bars or the left of vertical bars or boxes. Zero scores are retained. For distributions, the summary controls selection and ordering only; boxes use the fold values. Ranking direction is never inferred from a measure name. For smaller-is-better scores, use `rank_by = "signed", decreasing = FALSE`. Averaging fold p-values does not produce a combined p-value; choose a suitable summary outside this function when the score's interpretation requires it.

Set `bar_width` to a pixel thickness for separate zero-to-score segments. This uses native bars and covers the legacy importance `type = "line"` geometry without connecting different variables or adding a new chart type.

Explicit NA scores, omitted rows, and wholly unavailable folds are excluded from each variable's summary. Variables with no available score are omitted. A producer whose importance omits variables with a zero score supplies those zeros explicitly. Infinite scores are rejected. `summarize_varimp()` returns the summary and ranking this function draws.

Means and medians give each contributing fold equal weight. An incomplete summary describes the available folds, not an estimate guaranteed to be unbiased for all folds. Incomplete fold coverage is reported in the console; chart labels retain only the variable and measure names. The input is never modified.

Boxplots require fold-level records. Every available score is overlaid by default, with its fold ID in the tooltip. Missing values remain missing in the materialized table and are reported in the console. Boxes describe resample variability, not a confidence interval. See `draw_boxplot()` for quartiles, whiskers, and point placement. Use `whisker = 0` for full-range whiskers, as in the current live importance view.

For portable rendering, materialize the selected, summarized `label` and `importance` columns in display order and bind them using `setup_BarConfig(x = "label", y = "importance")`. Preserve variable IDs and contributing/total fold counts alongside these columns. Set `horizontal`, axis labels, and title explicitly. That table and config reproduce the view without a fitted model or R callbacks. The raw-data summary and selection options are not yet part of a shared visualization schema. For distributions, use one numeric column per selected variable and an observation column for the complete fold universe. Bind those columns and the variable labels through `setup_BoxplotConfig()`. Only selected variables are widened; selection does not discard folds needed to identify structural zeros.

Value

htmlwidget: ECharts importance bars or fold distributions.

Examples

```
draw_varimp(c(age = 0.8, weight = -0.4, height = 0.2))
scores <- data.frame(
  variable = c("age", "weight", "age"),
  fold = c("A", "A", "B"),
  gain = c(0.8, 0.4, 0.6)
```

```
)
draw_varimp(scores, measure = "gain")
draw_varimp(scores, measure = "gain", type = "boxplot")
```

`draw_violin` *Draw Violin Distributions*

Description

A convenience entry point for `draw_boxplot()` with violin geometry. Smoothing, grouping, observation overlays, paired lines, comparisons, and vector export follow the same contract.

Usage

```
draw_violin(x, ..., show_box = FALSE)
```

Arguments

`x`	Numeric or List: Numeric vectors, one per box; named lists supply category labels. With `group`, every vector must match its length.
`...`	Arguments passed to `draw_boxplot()`, except `geometry`.
`show_box`	Logical: Overlay a narrow box on each violin.

Value

htmlwidget: Vector violin distributions.

Examples

```
draw_violin(iris["Sepal.Length"], group = iris[["Species"]], show_box = TRUE)
```

`draw_volcano` *Draw a Volcano Plot*

Description

Plot signed estimates and raw p-values through SignificanceConfig. The Manhattan view places outcomes in input order; it does not use genomic coordinates. Missing pairs have no marks but remain in the testing family.

Usage

```
draw_volcano(
  x,
  pvals,
  xnames = NULL,
  p_adjust_method = "holm",
  p_transform = "neg_log10",
  p_thresh = 0.05,
  x_thresh = 0,
  annotate_n = 7L,
  ...,
  legend_position = "top",
  legend_placement = "outside",
  legend_title = NULL,
  theme = NULL,
  width = NULL,
  height = NULL,
  element_id = NULL,
  filename = NULL,
  group = NULL,
  zoom = FALSE
)
```

Arguments

x	Numeric vector: Signed estimates; NA values have no plotted mark.
pvals	Numeric vector: Raw p-values in [0, 1], or NA.
xnames	Optional Character vector: One outcome label per estimate.
p_adjust_method	Character: Method from stats::p.adjust.methods. Adjustment precedes transformation.
p_transform	Character {"neg_log10", "identity", "one_minus"}: Named display transform of adjusted p-values.
p_thresh	Numeric (0, 1]: Significance threshold on adjusted p-values; the comparison is strict.
x_thresh	Numeric: Effect threshold separating low and high significant results.
annotate_n	Integer [0, Inf): Maximum significant feature labels per side in the volcano view, ranked by raw p-value with input-order ties.
...	Additional named settings for setup_SignificanceConfig(), such as reference, zero_cap, n_tests, colors, axis limits, or margins.
legend_position	Character {"top", "bottom", "left", "right", "top-left", "top-right", "bottom-left", "bottom-right"}: Legend anchor. Top/bottom anchors use horizontal rows; left/right anchors use a vertical column. Corner anchors align within the top or bottom row.

`legend_placement`
: Character {"outside", "inside"}: Relation to the plotting area. Outside placement reserves space for the complete legend; inside placement overlays the data. Neither setting adds a missing legend.

`legend_title`
: Optional Character: Title drawn with the legend or colorbar: above left/right legends and vertical colorbars, before the first row of top/bottom legends and horizontal colorbars. `NULL` draws no title. A title does not add a missing legend.

`theme`
: Optional Theme: Widget/export theme.

`width, height`
: Optional Numeric or Character: Widget dimensions.

`element_id`
: Optional Character: HTML element identifier.

`filename`
: Optional Character: Static output path, currently SVG.

`group`
: Optional vector: One display-group label per outcome; unset uses significance and direction. Missing groups omit marks but retain the testing family.

`zoom`
: Logical: Enable Shift+wheel zoom and drag panning; see `setup_SignificanceConfig()`.

Value

An ECharts htmlwidget.

Statistical semantics

P-values are adjusted once over the full supplied family before rows with missing estimates or p-values are omitted from the marks. Missing hypotheses still count in `n_tests`; they do not become observed p-values of one. Raw and adjusted values remain in the compiled data. Function-valued transforms are not accepted by this portable configuration.

Under `neg_log10`, zero p-values have infinite mathematical height. They retain their zero value but are displayed at `zero_cap`, identified by triangles and a caption. No positive p-value is clamped. For identity and one-minus transforms, zeros are finite and no cap is used.

Examples

```
draw_volcano(c(-2, 0.1, 3), c(0.001, 0.6, 0.002),
  xnames = c("A", "B", "C"), p_adjust_method = "holm")
```

---

draw_xt *Draw Independent Time Series on Two Value Axes*

Description

Draw Independent Time Series on Two Value Axes

Usage

```
draw_xt(
  x,
  y,
  x2 = NULL,
  y2 = NULL,
  shade_interval = list(),
  shade_bin = NULL,
  zt = NULL,
  show_zt_every = 1L,
  ...,
  theme = NULL,
  width = NULL,
  height = NULL,
  element_id = NULL,
  filename = NULL
)
```

Arguments

x, y	Numeric, temporal vector, or named list: Primary time/value records.
x2, y2	Optional vector or list: Secondary records, with independent times. When x2 is unset, secondary values share x.
shade_interval	List: Increasing intervals on the x scale, in epoch milliseconds for Date/POSIXct axes. Applied using draw_annotate().
shade_bin	Optional Numeric: Zero/one shading flags aligned with x. Mutually exclusive with shade_interval; contiguous runs span their first and last observation. A single observation has zero interval width.
zt	Optional Numeric: Zeitgeber labels aligned with a single x vector.
show_zt_every	Integer [1, Inf): Keep every nth label, always including zero phase.
...	Additional settings passed to setup_TimeSeriesConfig().
theme	Optional Theme: Theme override. The palette inside the theme can be overridden per-chart with the color argument.
width	Optional Character or Numeric: Widget width.
height	Optional Character or Numeric: Widget height.
element_id	Optional Character: Explicit DOM element ID for the widget container. NULL lets htmlwidgets generate one.
filename	Optional Character: If provided, save the widget to this file via save_drawing().

Value

htmlwidget: Independent series sharing the time axis.

Examples

```
draw_xt(1:5, list(Temperature = 20:24), y2 = list(Pressure = 100:104))
```

EChartsOption *ECharts Option*

Description

The top-level configuration object passed to `echarts.setOption()`. Holds all chart components, series, and global settings.

Usage

```
EChartsOption(
  title = NULL,
  legend = NULL,
  x_axis_3d = NULL,
  y_axis_3d = NULL,
  z_axis_3d = NULL,
  grid_3d = NULL,
  grid = NULL,
  x_axis = NULL,
  y_axis = NULL,
  tooltip = NULL,
  visual_map = NULL,
  graphic = NULL,
  toolbox = NULL,
  data_zoom = NULL,
  series = NULL,
  color = NULL,
  background_color = NULL,
  text_style = NULL,
  animation = NULL,
  animation_threshold = NULL,
  animation_duration = NULL,
  animation_easing = NULL,
  animation_delay = NULL,
  animation_duration_update = NULL,
  dark_mode = NULL,
  use_utc = NULL
)
```

Arguments

`title`	Optional Title or list: Title configuration.
`legend`	Optional Legend or list: Legend configuration.
`x_axis_3d`, `y_axis_3d`, `z_axis_3d`, `grid_3d`	Optional List: ECharts-GL cartesian axes and grid, as defined by its Axis3D and Grid3DModel sources.

`grid` Optional Grid or list: Grid configuration.

`x_axis` Optional Axis or list: X-axis configuration (`xAxis`).

`y_axis` Optional Axis or list: Y-axis configuration (`yAxis`).

`tooltip` Optional Tooltip: Tooltip configuration.

`visual_map` Optional VisualMap or list: Visual map configuration (`visualMap`). Required for heatmaps to map data values to colors.

`graphic` Optional list: Native graphic annotations, corresponding to `GraphicComponentOption` in `src/component/graphic/GraphicModel.ts`.

`toolbox` Optional list: Toolbox configuration (`toolbox`), e.g. the `dataZoom` and `restore` features used by `draw_gantt()`.

`data_zoom` Optional DataZoom or list of DataZoom: Axis zoom component(s) (`dataZoom`). Pass a list to combine, e.g., a `"slider"` with an `"inside"` zoom.

`series` Optional series object or list: Series configuration (LineSeries, BarSeries, ScatterSeries, PieSeries, BoxplotSeries, HeatmapSeries).

`color` Optional Character: Color palette.

`background_color` Optional Character: Chart background color.

`text_style` Optional TextStyle: Global default text style.

`animation` Optional Logical: Whether to enable animation.

`animation_threshold` Optional Numeric `[0, Inf)`: Data-count threshold that disables animation.

`animation_duration` Optional Numeric `[0, Inf)`: Initial animation duration in milliseconds.

`animation_easing` Optional Character: Animation easing function name.

`animation_delay` Optional Numeric `[0, Inf)`: Initial animation delay in milliseconds.

`animation_duration_update` Optional Numeric `[0, Inf)`: Duration in milliseconds of the transition on data or zoom updates.

`dark_mode` Optional Logical or Character {"auto"}: Dark-mode setting.

`use_utc` Optional Logical: Whether to use UTC for time axes.

Details

Corresponds to `EChartsOption` in `src/export/option.ts` (line 258) and `ECUnitOption` in `src/util/types.ts` (line 697).

Value

`EChartsOption` object.

Examples

```
EChartsOption(
  title = Title(text = "Weekly sessions"),
  x_axis = Axis(type = "category", data = c("Mon", "Tue", "Wed")),
  y_axis = Axis(type = "value"),
  series = LineSeries(data = c(150, 230, 224), smooth = TRUE)
)
```

FitLayer *Supplied Fit Layer*

Description

Coordinates and optional interval bounds for a fitted curve, independent of the model or language that produced it. Uses LineSeriesOption and CustomSeriesOption in ECharts src/chart/line and src/chart/custom. Rows are sorted together by x. Missing predictions must be removed before construction; use separate layers for disconnected intervals. No fitting is performed here.

Usage

```
FitLayer(
  x = c(0, 1),
  y = c(0, 1),
  lower = NULL,
  upper = NULL,
  name = "Fit",
  color = "#6ca3a0",
  line_width = 2,
  opacity = 0.2,
  x_axis = 0L,
  y_axis = 0L
)
```

Arguments

x, y	Numeric: Finite curve coordinates with distinct x values.
lower, upper	Optional Numeric: Paired finite interval bounds at every x.
name	Character: Legend label; a matching observation label links selection.
color	Character: Curve and interval color.
line_width	Numeric (0, Inf): Curve width in pixels.
opacity	Numeric [0, 1]: Interval opacity.
x_axis, y_axis	Integer [0, Inf): Zero-based axis indices.

Value

FitLayer: Validated, serializable prediction geometry.

Examples

```
FitLayer(x = 1:3, y = c(2, 4, 5), lower = c(1, 3, 4), upper = c(3, 5, 6))
```

`GanttConfig` *Gantt Chart Configuration*

Description

A serializable description of a Gantt chart. Build one with `setup_GanttConfig()` rather than calling this constructor directly.

Usage

```
GanttConfig(
  dat_path = NULL,
  title = NULL,
  origin = NULL,
  writer = NULL,
  legend_position = "top",
  legend_placement = "outside",
  legend_title = NULL,
  label = "label",
  start = "start",
  end = "end",
  group = NULL,
  tooltip = NULL,
  border = NULL,
  axis_type = "value",
  zoom = TRUE,
  guides = TRUE,
  bar_height = 0.6,
  bar_radius = 0,
  border_color = "#E53935",
  border_width = 1.5,
  palette = NULL,
  xlab = NULL
)
```

Arguments

`dat_path`	Optional Character: Path to the data, read at draw time. The serializable alternative to passing `data` to `draw()`.
`title`	Optional Character: Chart title. Declared here because it is the one property every chart type has; `palette` is not (four take a `colormap` instead), and axis labels and margins are cartesian-only.

`origin` Optional Named character {"user", "default", "derived"}: Where each value came from, one entry per settable property. Absent on an authored config; written by the interface that resolved it.

`writer` Optional Named character: Which interface wrote the config, as `name` and `version`. Absent on an authored config.

`legend_position` Character {"top", "bottom", "left", "right", "top-left", "top-right", "bottom-left", "bottom-right"}: Legend anchor. Top/bottom anchors use horizontal rows; left/right anchors use a vertical column. Corner anchors align within the top or bottom row.

`legend_placement` Character {"outside", "inside"}: Relation to the plotting area. Outside placement reserves space for the complete legend; inside placement overlays the data. Neither setting adds a missing legend.

`legend_title` Optional Character: Title drawn with the legend or colorbar: above left/right legends and vertical colorbars, before the first row of top/bottom legends and horizontal colorbars. `NULL` draws no title. A title does not add a missing legend.

`label` Character: Column naming each task.

`start` Character: Column holding each task's start.

`end` Character: Column holding each task's end.

`group` Optional Character: Column to color bars by.

`tooltip` Optional Character: Column holding per-bar tooltip text.

`border` Optional Character: Logical column; `TRUE` outlines the bar.

`axis_type` Character {"value", "time"}: How the time axis is scaled.

`zoom` Logical: Enable Shift+wheel zoom, drag panning, and box-zoom controls on both axes.

`guides` Logical: Show guide lines.

`bar_height` Numeric `[0, 1]`: Bar height as a fraction of the row.

`bar_radius` Numeric `[0, Inf)`: Bar corner radius in pixels.

`border_color` Character: Outline color for bordered bars.

`border_width` Numeric `[0, Inf)`: Outline width in pixels.

`palette` Optional Character: Bar colors, overriding the theme palette for this chart. `NULL` uses the theme's.

`xlab` Optional Character: Time axis label.

Details

The bound data is a table of tasks, one row per bar. `label`, `start` and `end` name its required columns; `group`, `tooltip` and `border` name optional ones.

Value

`GanttConfig` object.

Author(s)

EDG

Examples

```
setup_GanttConfig()@type
```

GraphEdge *Graph Edge*

Description

One edge (link) in a network. The R analog of the GraphEdge TypeScript interface in rtemislive (~/Code/live/src/lib/types.ts).

Usage

```
GraphEdge(
  source = character(0),
  target = character(0),
  weight = NULL,
  sign = NULL
)
```

Arguments

source	Character: Source node id.
target	Character: Target node id.
weight	Optional Numeric: Edge magnitude; drives thickness. For a correlation graph, the absolute value of r.
sign	Optional Numeric {-1, 1}: Sign of the underlying value (e.g. correlation): 1 positive, -1 negative. Drives edge color when present.

Value

GraphEdge object.

Examples

```
GraphEdge(source = "a", target = "b", weight = 0.8)
```

GraphModel *Graph Model*

Description

A complete, renderer-agnostic network: a list of GraphNode objects, a list of GraphEdge objects, and a directedness flag. The R analog of the `GraphModel` TypeScript interface in rtemislive (`~/Code/live/src/lib/types.ts`), consumed by the `rtemis-graph` htmlwidget.

Usage

```
GraphModel(nodes = list(), edges = list(), directed = logical(0))
```

Arguments

`nodes`	List: List of GraphNode objects.
`edges`	List: List of GraphEdge objects.
`directed`	Logical: Whether edges are directed.

Details

Most users build this implicitly through `draw_network()`; the constructor is exported for power users who assemble nodes and edges directly.

Value

`GraphModel` object.

Examples

```
GraphModel(
  nodes = list(GraphNode(id = "a"), GraphNode(id = "b")),
  edges = list(GraphEdge(source = "a", target = "b")),
  directed = FALSE
)
```

GraphNode *Graph Node*

Description

One node (vertex) in a network. The R analog of the `GraphNode` TypeScript interface in rtemislive (`~/Code/live/src/lib/types.ts`).

Usage

```
GraphNode(id = character(0), label = NULL, value = NULL, group = NULL)
```

Arguments

id	Character: Unique node identifier. Edges reference nodes by this id.
label	Optional Character: Display label. Defaults to id in the renderer when absent.
value	Optional Numeric: Drives node size when scale_by_degree is enabled (e.g. weighted degree).
group	Optional Character: Categorical key for node color.

Value

GraphNode object.

Examples

```
GraphNode(id = "a", label = "Node A", group = "g1")
```

Grid *Grid*

Description

Configuration for the cartesian coordinate grid container.

Usage

```
Grid(
  show = NULL,
  left = NULL,
  right = NULL,
  top = NULL,
  bottom = NULL,
  width = NULL,
  height = NULL,
  contain_label = NULL,
  background_color = NULL,
  border_width = NULL,
  border_color = NULL,
  shadow_blur = NULL,
  shadow_color = NULL,
  shadow_offset_x = NULL,
  shadow_offset_y = NULL
)
```

Arguments

show	Optional Logical: Whether to show the grid border.
left	Optional Numeric or Character: Distance from the left side of the container.
right	Optional Numeric or Character: Distance from the right side of the container.
top	Optional Numeric or Character: Distance from the top of the container.
bottom	Optional Numeric or Character: Distance from the bottom of the container.
width	Optional Numeric or Character: Grid width.
height	Optional Numeric or Character: Grid height.
contain_label	Optional Logical: Whether the grid region contains axis labels.
background_color	Optional Character: Grid background color.
border_width	Optional Numeric [0, Inf): Grid border width.
border_color	Optional Character: Grid border color.
shadow_blur	Optional Numeric [0, Inf): Shadow blur radius.
shadow_color	Optional Character: Shadow color.
shadow_offset_x	Optional Numeric: Shadow horizontal offset.
shadow_offset_y	Optional Numeric: Shadow vertical offset.

Details

Corresponds to GridOption in src/coord/cartesian/GridModel.ts (line 37). ECharts docs: https://echarts.apache.org/en/option.html#grid

Value

Grid object.

Examples

```
Grid(left = "10%", right = "10%", top = 60, bottom = 60, contain_label = TRUE)
```

HeatmapConfig *Heatmap Configuration*

Description

A serializable description of a heatmap. Build one with setup_HeatmapConfig() rather than calling this constructor directly.

Usage

```
HeatmapConfig(
  dat_path = NULL,
  title = NULL,
  origin = NULL,
  writer = NULL,
  legend_position = "right",
  legend_placement = "outside",
  legend_title = NULL,
  row_names = NULL,
  col_names = NULL,
  triangle = NULL,
  cluster_rows = FALSE,
  cluster_cols = FALSE,
  dist_method = "euclidean",
  hclust_method = "complete",
  square_cells = NULL,
  show_row_dendro = TRUE,
  show_col_dendro = TRUE,
  dendro_row_width = 60,
  dendro_col_height = 60,
  dendro_color = NULL,
  dendro_uniform = FALSE,
  dendro_row_side = "right",
  dendro_col_side = "top",
  colormap = NULL,
  zlim = NULL,
  show_notes = FALSE,
  row_cut = NULL,
  col_cut = NULL,
  show_values = FALSE,
  value_digits = 2L,
  show_colorbar = TRUE,
  colorbar_orient = "vertical",
  margin_top = NULL,
  margin_right = NULL,
  margin_bottom = NULL,
  margin_left = NULL
)
```

Arguments

dat_path	Optional Character: Path to the data, read at draw time. The serializable alternative to passing data to draw().
title	Optional Character: Chart title. Declared here because it is the one property every chart type has; palette is not (four take a colormap instead), and axis labels and margins are cartesian-only.

`origin` Optional Named character {"user", "default", "derived"}: Where each value came from, one entry per settable property. Absent on an authored config; written by the interface that resolved it.

`writer` Optional Named character: Which interface wrote the config, as `name` and `version`. Absent on an authored config.

`legend_position`
Character {"top", "bottom", "left", "right", "top-left", "top-right", "bottom-left", "bottom-right"}: Legend anchor. Top/bottom anchors use horizontal rows; left/right anchors use a vertical column. Corner anchors align within the top or bottom row.

`legend_placement`
Character {"outside", "inside"}: Relation to the plotting area. Outside placement reserves space for the complete legend; inside placement overlays the data. Neither setting adds a missing legend.

`legend_title` Optional Character: Title drawn with the legend or colorbar: above left/right legends and vertical colorbars, before the first row of top/bottom legends and horizontal colorbars. `NULL` draws no title. A title does not add a missing legend.

`row_names`, `col_names`
Optional Character: Labels overriding the matrix dimnames.

`triangle` Optional Character {"lower", "upper"}: Show only one triangle, for a symmetric matrix.

`cluster_rows`, `cluster_cols`
Logical: Reorder by hierarchical clustering.

`dist_method` Character: Distance measure for clustering.

`hclust_method` Character: Linkage method for clustering.

`square_cells` Optional Logical: Force square cells, sizing the container to match.

`show_row_dendro`, `show_col_dendro`
Logical: Draw each dendrogram.

`dendro_row_width`, `dendro_col_height`
Numeric `[0, Inf)`: Dendrogram panel size in pixels.

`dendro_color` Optional Character: Dendrogram line color.

`dendro_uniform` Logical: Draw dendrograms with uniform branch heights.

`dendro_row_side`
Character {"left", "right"}: Row dendrogram side.

`dendro_col_side`
Character {"top", "bottom"}: Column dendrogram side.

`colormap` Optional Character: Colors defining the continuous scale. `NULL` derives a theme-aware diverging or sequential scale.

`zlim` Optional Numeric: Color-scale limits, length 2.

`show_notes` Logical: Print supplied cell notes instead of numeric values.

`row_cut`, `col_cut`
Optional Integer: Number of clusters for branch coloring, requiring a computed or supplied tree.

`show_values` Logical: Print each cell's value.

`value_digits` Integer `[0, Inf)`: Digits for printed values.

`show_colorbar` Logical: Draw the color bar.

`colorbar_orient`
Character {"vertical", "horizontal"}: Color bar orientation.

`margin_top`, `margin_right`, `margin_bottom`, `margin_left`
Optional Integer `[0, Inf)`: Override the auto-computed margins, per side.

Details

The bound data is a numeric matrix, or a record containing `values` and optional `row_tree`, `col_tree`, `cell_notes`, `row_colors`, and `col_colors` (see `draw_heatmap()`). These annotations remain data, outside the config.

For a matrix, `row_names` and `col_names` override the matrix's own dimnames rather than naming columns to read.

Value

`HeatmapConfig` object.

Author(s)

EDG

Examples

```
setup_HeatmapConfig(cluster_rows = TRUE)@type
```

`HeatmapSeries` *Heatmap Series*

Description

Configuration for a heatmap chart series on Cartesian coordinates. Each data point is a triple `[col_index, row_index, value]` and color encoding is driven by a VisualMap component.

Usage

```
HeatmapSeries(
  name = NULL,
  data = NULL,
  x_axis_index = NULL,
  y_axis_index = NULL,
  silent = NULL,
  legend_hover_link = NULL,
  z_level = NULL,
  z = NULL,
```

```
  label = NULL,
  item_style = NULL,
  emphasis = NULL
)
```

Arguments

name	Optional Character: Series name.
data	Optional list: Heatmap data. Each element is a length-3 list `list(col_index, row_index, value)`.
x_axis_index	Optional Numeric `[0, Inf)`: X-axis index.
y_axis_index	Optional Numeric `[0, Inf)`: Y-axis index.
silent	Optional Logical: Whether the series is silent (no events).
legend_hover_link	Optional Logical: Whether to link hover with the legend.
z_level	Optional Numeric: Canvas layer index.
z	Optional Numeric: Front-back order within the same layer.
label	Optional LabelOption: Cell label configuration (e.g. show the value).
item_style	Optional ItemStyle: Cell styling (e.g. border color/width).
emphasis	Optional list: Emphasis (hover) styling. A plain list passed through to ECharts, e.g. `list(itemStyle = list(shadowBlur = 10))`.

Details

Corresponds to `HeatmapSeriesOption` in `src/chart/heatmap/HeatmapSeries.ts`. ECharts docs: https://echarts.apache.org/en/option.html#series-heatmap

Value

`HeatmapSeries` object.

Examples

```
# Each element is one cell, as c(column index, row index, value).
HeatmapSeries(name = "Density", data = list(c(0, 0, 5), c(1, 0, 3)))
```

HistogramConfig *Histogram Configuration*

Description

A serializable description of a histogram. Build one with `setup_HistogramConfig()` rather than calling this constructor directly.

Usage

```
HistogramConfig(
  dat_path = NULL,
  title = NULL,
  origin = NULL,
  writer = NULL,
  legend_position = "top",
  legend_placement = "outside",
  legend_title = NULL,
  mode = "overlap",
  order = "input",
  n = 512L,
  bw = "nrd0",
  bandwidth = NULL,
  kernel = "gaussian",
  adjust = 1,
  na_rm = TRUE,
  x = NULL,
  group = NULL,
  breaks = "Sturges",
  bins = NULL,
  bin_edges = NULL,
  normalization = "count",
  bin_stat = "count",
  bar_mode = "overlay",
  zoom = FALSE,
  density = FALSE,
  fill_alpha = NULL,
  border_alpha = 1,
  palette = NULL,
  xlab = NULL,
  ylab = NULL,
  margin_top = NULL,
  margin_right = NULL,
  margin_bottom = NULL,
  margin_left = NULL
)
```

Arguments

dat_path	Optional Character: Path to the data, read at draw time. The serializable alternative to passing `data` to `draw()`.
title	Optional Character: Chart title. Declared here because it is the one property every chart type has; `palette` is not (four take a `colormap` instead), and axis labels and margins are cartesian-only.
origin	Optional Named character {"user", "default", "derived"}: Where each value came from, one entry per settable property. Absent on an authored config; written by the interface that resolved it.

`writer` Optional Named character: Which interface wrote the config, as `name` and `version`. Absent on an authored config.

`legend_position`
Character {"top", "bottom", "left", "right", "top-left", "top-right", "bottom-left", "bottom-right"}: Legend anchor. Top/bottom anchors use horizontal rows; left/right anchors use a vertical column. Corner anchors align within the top or bottom row.

`legend_placement`
Character {"outside", "inside"}: Relation to the plotting area. Outside placement reserves space for the complete legend; inside placement overlays the data. Neither setting adds a missing legend.

`legend_title` Optional Character: Title drawn with the legend or colorbar: above left/right legends and vertical colorbars, before the first row of top/bottom legends and horizontal colorbars. `NULL` draws no title. A title does not add a missing legend.

`mode` Character `{"overlap", "ridge"}`: Overlay samples or align rows on common x and y scales. Ridge rows use sample names in place of a legend. Increase figure height for many rows.

`order` Character `{"input", "mean", "median"}`: Sample order. Summary orders are decreasing, retain ties in input order, and place empty samples last.

`n` Integer `[2, Inf)`: Number of density evaluation points.

`bw` Character or Numeric: Bandwidth selector (`"nrd0"`, `"nrd"`, `"ucv"`, `"bcv"`, `"SJ"`, `"SJ-ste"`, `"SJ-dpi"`). Numeric input is normalized to `bandwidth` by the setup function.

`bandwidth` Optional Numeric `(0, Inf)`: Explicit finite smoothing bandwidth in measurement units, overriding the selector.

`kernel` Character: One of `"gaussian"`, `"epanechnikov"`, `"rectangular"`, `"triangular"`, `"biweight"`, `"cosine"`, or `"optcosine"`.

`adjust` Numeric `(0, Inf)`: Finite bandwidth multiplier.

`na_rm` Logical: Whether to remove `NA` values before computing densities.

`x` Optional Character: Columns to bin, one distribution per column.

`group` Optional Character: Column to split the bins by.

`breaks` Numeric, Character, or Numeric vector: Binning method. A single number (number of bins), a character string naming an algorithm (e.g. `"Sturges"`, `"Scott"`, `"FD"`), or a numeric vector of break points. Numeric forms are normalized to `bins` or `bin_edges` by the setup function. Bin counts are suggestions to `hist()`.

`bins` Optional Integer `[1, 1000000]`: Suggested bin count.

`bin_edges` Optional Numeric vector: Finite, strictly increasing edges spanning every retained observation. Overrides `breaks`; excludes `bins`.

`normalization` Character: `"count"`, `"probability"`, `"percent"`, `"density"`, or `"count_density"`.

`bin_stat` Character `{"count", "sum", "mean", "min", "max"}`: Statistic of x observations in each bin. Empty bins have height zero. Non-count statistics require normalization = "count" and density = FALSE.

bar_mode	Character {"overlay", "group", "stack"}: Histogram group layout. Stacks accumulate positive and negative values separately. Group/stack layouts require mode = "overlap" and density = FALSE.
zoom	Logical: Enable a slider, Shift+wheel zoom, and drag panning on the x-axis. Ridge rows zoom together, without a slider.
density	Logical: Overlay a kernel density in histogram units.
fill_alpha	Optional Numeric [0, 1]: Bar fill opacity. NULL uses 0.5 when overlaid groups overlap (bar_mode = "overlay" with more than one nonempty group outside ridge mode) and 0.75 otherwise.
border_alpha	Numeric [0, 1]: Opacity of the bar borders, drawn in the series color. 0 draws no border.
palette	Optional Character: Series colors, overriding the theme palette for this chart. NULL uses the theme's.
xlab, ylab	Optional Character: Axis labels. NULL derives them from the data.
margin_top, margin_right, margin_bottom, margin_left	Optional Integer [0, Inf): Plot margins in pixels.

Details

x names one or more columns to bin. group names a column to split them by, drawing one series per level. The output uses CustomSeriesOption in ECharts src/chart/custom/CustomSeries.ts and LineSeriesOption in src/chart/line/LineSeries.ts for optional density overlays.

Value

HistogramConfig object.

Author(s)

EDG

Examples

```
setup_HistogramConfig(x = "mpg")@type
```

ItemStyle *Item Style*

Description

Styling for graphical items (bar segments, scatter points, pie slices, etc.). Includes both fill and stroke options.

Usage

```
ItemStyle(
  color = NULL,
  opacity = NULL,
  border_color = NULL,
  border_width = NULL,
  border_type = NULL,
  border_cap = NULL,
  border_join = NULL,
  border_dash_offset = NULL,
  border_miter_limit = NULL,
  border_radius = NULL,
  decal = NULL,
  shadow_blur = NULL,
  shadow_color = NULL,
  shadow_offset_x = NULL,
  shadow_offset_y = NULL
)
```

Arguments

color	Optional Character: CSS fill color.
opacity	Optional Numeric [0, 1]: Fill opacity.
border_color	Optional Character: Border color.
border_width	Optional Numeric [0, Inf): Border width in pixels.
border_type	Optional Character {"solid", "dashed", "dotted"}: Border line type.
border_cap	Optional Character {"butt", "round", "square"}: Border line cap.
border_join	Optional Character {"bevel", "round", "miter"}: Border line join.
border_dash_offset	Optional Numeric: Border dash offset.
border_miter_limit	Optional Numeric [0, Inf): Border miter limit.
border_radius	Optional Numeric, Character, or Numeric vector: Corner border radius.
decal	Optional Any: Decal pattern for accessibility.
shadow_blur	Optional Numeric [0, Inf): Shadow blur radius.
shadow_color	Optional Character: Shadow color.
shadow_offset_x	Optional Numeric: Shadow horizontal offset.
shadow_offset_y	Optional Numeric: Shadow vertical offset.

Details

Corresponds to ItemStyleOption in src/util/types.ts (line 1142). ECharts docs: https://echarts.apache.org/en/option.html#series-bar.itemStyle

Value

ItemStyle object.

Examples

```
ItemStyle(color = "#16a085", border_color = "#ffffff", border_width = 1)
```

LabelLine *Label Line*

Description

Connecting line between a label and its chart element (e.g. pie label lines).

Usage

```
LabelLine(
  show = NULL,
  show_above = NULL,
  length = NULL,
  length2 = NULL,
  smooth = NULL,
  min_turn_angle = NULL,
  line_style = NULL
)
```

Arguments

show	Optional Logical: Whether to show the label line.
show_above	Optional Logical: Whether the line renders above other elements.
length	Optional Numeric [0, Inf): Length of the first segment.
length2	Optional Numeric [0, Inf): Length of the second segment.
smooth	Optional Numeric [0, 1] or Logical: Line smoothness.
min_turn_angle	Optional Numeric: Minimum angle for turning the line.
line_style	Optional LineStyle: Line styling.

Details

Corresponds to LabelLineOption in src/util/types.ts (line 1379). ECharts docs: https://echarts.apache.org/en/option.html#series-pie.labelLine

Value

LabelLine object.

Examples

```
LabelLine(show = TRUE, length = 10, length2 = 15)
```

`LabelOption` *Label Option*

Description

Text label configuration for chart elements. Combines label-specific positioning (show, position, distance, rotate) with text styling.

Usage

```
LabelOption(
  show = NULL,
  position = NULL,
  distance = NULL,
  rotate = NULL,
  offset = NULL,
  formatter = NULL,
  silent = NULL,
  precision = NULL,
  value_animation = NULL,
  min_margin = NULL,
  text_style = NULL
)
```

Arguments

`show`	Optional Logical: Whether to show the label.
`position`	Optional Character {"top", "left", "right", "bottom", "inside", "insideLeft", "insideRight", "insideTop", "insideBottom", "insideTopLeft", "insideBottomLeft", "insideTopRight", "insideBottomRight", "outside"} or Numeric: Label position. `"bottom"`, `"inside"`, `"insideLeft"`, `"insideRight"`, `"insideTop"`, `"insideBottom"`, `"insideTopLeft"`, `"insideBottomLeft"`, `"insideTopRight"`, `"insideBottomRight"`, `"outside"`.
`distance`	Optional Numeric: Distance from the element.
`rotate`	Optional Numeric: Rotation angle in degrees.
`offset`	Optional Numeric: Offset `c(x, y)` in pixels.
`formatter`	Optional Character or function: Label text formatter.
`silent`	Optional Logical: Whether the label is silent.
`precision`	Optional Numeric or Character {"auto"}: Numeric precision.
`value_animation`	Optional Logical: Whether to animate value changes.
`min_margin`	Optional Numeric `[0, Inf)`: Minimum margin between labels.
`text_style`	Optional TextStyle: Text appearance.

Details

Corresponds to SeriesLabelOption / LabelOption in src/util/types.ts (line 1283, 1348). ECharts docs: https://echarts.apache.org/en/option.html#series-bar.label

Value

LabelOption object.

Examples

```
LabelOption(show = TRUE, position = "top", formatter = "{c}")
```

Legend *Legend*

Description

Legend configuration for selecting/filtering series.

Usage

```
Legend(
  show = NULL,
  orient = NULL,
  align = NULL,
  left = NULL,
  right = NULL,
  top = NULL,
  bottom = NULL,
  width = NULL,
  height = NULL,
  padding = NULL,
  item_gap = NULL,
  item_width = NULL,
  item_height = NULL,
  icon = NULL,
  selected_mode = NULL,
  selected = NULL,
  formatter = NULL,
  inactive_color = NULL,
  inactive_border_color = NULL,
  text_style = NULL,
  item_style = NULL,
  background_color = NULL,
  border_color = NULL,
  border_width = NULL,
  border_radius = NULL,
```

```
  trigger_event = NULL,
  data = NULL
)
```

Arguments

show	Optional Logical: Whether to show the legend.
orient	Optional Character {"horizontal", "vertical"}: Layout orientation.
align	Optional Character {"auto", "left", "right"}: Marker and text alignment.
left	Optional Numeric or Character: Left position.
right	Optional Numeric or Character: Right position.
top	Optional Numeric or Character: Top position.
bottom	Optional Numeric or Character: Bottom position.
width	Optional Numeric or Character: Legend box width.
height	Optional Numeric or Character: Legend box height.
padding	Optional Numeric: Padding inside the legend box.
item_gap	Optional Numeric `[0, Inf)`: Gap between legend items.
item_width	Optional Numeric `[0, Inf)`: Legend symbol width.
item_height	Optional Numeric `[0, Inf)`: Legend symbol height.
icon	Optional Character: Default legend icon.
selected_mode	Optional Logical or Character {"single", "multiple"}: Selection mode.
selected	Optional Named logical vector: Pre-selected items.
formatter	Optional Character or function: Label formatter.
inactive_color	Optional Character: Color for unselected items.
inactive_border_color	Optional Character: Border color for unselected items.
text_style	Optional TextStyle: Legend label text style.
item_style	Optional ItemStyle: Legend icon style.
background_color	Optional Character: Background color.
border_color	Optional Character: Border color.
border_width	Optional Numeric `[0, Inf)`: Border width.
border_radius	Optional Numeric: Border radius.
trigger_event	Optional Logical: Whether the legend emits events.
data	Optional Character vector or list: Legend data items.

Details

Corresponds to `LegendOption` in `src/component/legend/LegendModel.ts` (line 162). ECharts docs: https://echarts.apache.org/en/option.html#legend

Value

Legend object.

Examples

```
Legend(show = TRUE, orient = "vertical", left = "left")
```

`LineConfig` *Line Chart Configuration*

Description

A serializable description of a line chart. Build one with `setup_LineConfig()` rather than calling this constructor directly.

Usage

```
LineConfig(
  dat_path = NULL,
  title = NULL,
  origin = NULL,
  writer = NULL,
  legend_position = "top",
  legend_placement = "outside",
  legend_title = NULL,
  x = NULL,
  y = NULL,
  group = NULL,
  blocks = NULL,
  smooth = FALSE,
  area = FALSE,
  points = TRUE,
  zoom = FALSE,
  palette = NULL,
  block_color = NULL,
  block_opacity = 0.2,
  square = FALSE,
  equal_axes = FALSE,
  pad = 0.04,
  xlim = NULL,
  ylim = NULL,
  xlab = NULL,
  ylab = NULL,
  margin_top = NULL,
  margin_right = NULL,
  margin_bottom = NULL,
  margin_left = NULL
)
```

Arguments

dat_path	Optional Character: Path to the data, read at draw time. The serializable alternative to passing data to draw().
title	Optional Character: Chart title. Declared here because it is the one property every chart type has; palette is not (four take a colormap instead), and axis labels and margins are cartesian-only.
origin	Optional Named character {"user", "default", "derived"}: Where each value came from, one entry per settable property. Absent on an authored config; written by the interface that resolved it.
writer	Optional Named character: Which interface wrote the config, as name and version. Absent on an authored config.
legend_position	Character {"top", "bottom", "left", "right", "top-left", "top-right", "bottom-left", "bottom-right"}: Legend anchor. Top/bottom anchors use horizontal rows; left/right anchors use a vertical column. Corner anchors align within the top or bottom row.
legend_placement	Character {"outside", "inside"}: Relation to the plotting area. Outside placement reserves space for the complete legend; inside placement overlays the data. Neither setting adds a missing legend.
legend_title	Optional Character: Title drawn with the legend or colorbar: above left/right legends and vertical colorbars, before the first row of top/bottom legends and horizontal colorbars. NULL draws no title. A title does not add a missing legend.
x	Optional Character: Column along the horizontal axis. A numeric column gets a value axis, a Date or POSIXct column a time axis with adaptively chosen labels, and any other column a category axis.
y	Optional Character: Columns to plot, one line each.
group	Optional Character: Column to group and color lines by. Splits a single y column into one line per level; cannot be combined with several y columns or with blocks.
blocks	Optional Character: Column whose contiguous runs shade vertical background bands.
smooth	Logical: Draw smoothed lines.
area	Logical: Fill the area under each line.
points	Logical: Show a marker at each data value.
zoom	Logical: Enable an x-axis slider, Shift+wheel zoom, and drag panning.
palette	Optional Character: Series colors, overriding the theme palette for this chart. NULL uses the theme's.
block_color	Optional Character: Band colors.
block_opacity	Numeric [0, 1]: Band opacity.
square	Logical: If TRUE, draw the plotting box square – equal height and width in pixels, excluding axis labels and margins.

`equal_axes`	Logical: If TRUE, give one data unit the same size in pixels on both axes. Set with `square` for a plot that is both, such as an ROC curve; the two axes are then made to span the same interval. Requires a numeric x.
`pad`	Numeric `[0, Inf)`: Fraction of the data range to extend each axis by when the limits are not given.
`xlim`, `ylim`	Optional Numeric: Axis limits, length 2. `NULL` derives them from the data, padded by `pad`. On a time axis `xlim` is in epoch milliseconds, the unit the chart positions by.
`xlab`, `ylab`	Optional Character: Axis labels. `NULL` derives them from the data.
`margin_top`, `margin_right`, `margin_bottom`, `margin_left`	Optional Integer `[0, Inf)`: Plot margins in pixels.

Details

`x` names the column along the horizontal axis. `y` names **one or more** columns, one line each, as BarConfig does.

Value

`LineConfig` object.

Author(s)

EDG

Examples

```
setup_LineConfig(x = "wt", y = "mpg")@type
```

`LineSeries` *Line Series*

Description

Configuration for a line chart series.

Usage

```
LineSeries(
  name = NULL,
  data = NULL,
  x_axis_index = NULL,
  y_axis_index = NULL,
  stack = NULL,
  silent = NULL,
  legend_hover_link = NULL,
  color = NULL,
```

```
  z_level = NULL,
  z = NULL,
  smooth = NULL,
  step = NULL,
  connect_nulls = NULL,
  clip = NULL,
  show_symbol = NULL,
  symbol = NULL,
  symbol_size = NULL,
  symbol_rotate = NULL,
  symbol_offset = NULL,
  line_style = NULL,
  area_style = NULL,
  item_style = NULL,
  label = NULL,
  mark_area = NULL
)
```

Arguments

`name`	Optional Character: Series name for legend display.
`data`	Optional Numeric or list: Data values.
`x_axis_index`	Optional Numeric `[0, Inf)`: X-axis index.
`y_axis_index`	Optional Numeric `[0, Inf)`: Y-axis index.
`stack`	Optional Character: Stack group name.
`silent`	Optional Logical: Whether the series is silent.
`legend_hover_link`	Optional Logical: Whether to highlight related legends on hover.
`color`	Optional Character: Series color override.
`z_level`	Optional Numeric: Canvas layer index.
`z`	Optional Numeric: Front-back order within the same layer.
`smooth`	Optional Logical or Numeric `[0, 1]`: Line smoothing.
`step`	Optional Logical or Character {"start", "end", "middle"}: Step line mode.
`connect_nulls`	Optional Logical: Whether to connect across `NULL` points.
`clip`	Optional Logical: Whether to clip overflow.
`show_symbol`	Optional Logical: Whether to show data-point symbols.
`symbol`	Optional Character {"circle", "rect", "roundRect", "triangle", "diamond", "pin", "arrow", "none"}: Symbol type. `"diamond"`, `"pin"`, `"arrow"`, `"none"`, or custom SVG path.
`symbol_size`	Optional Numeric: Symbol size in pixels.
`symbol_rotate`	Optional Numeric: Symbol rotation in degrees, counter-clockwise-positive. A symbol is unrotated at `0`.

symbol_offset	Optional Numeric or Character: Shift from the data point as c(x, y), in screen pixels (numbers) or as a percentage of the symbol's own size (strings, e.g. c("0%", "-50%")). Applied *after* symbol_rotate, in the unrotated frame.
line_style	Optional LineStyle: Line styling.
area_style	Optional AreaStyle: Area fill styling.
item_style	Optional ItemStyle: Data-point marker styling.
label	Optional LabelOption: Data-label configuration.
mark_area	Optional MarkArea: Shaded bands (e.g. vertical highlight regions across x-ranges).

Details

Corresponds to LineSeriesOption in src/chart/line/LineSeries.ts (line 74). ECharts docs: https://echarts.apache.org/en/option.html#series-line

Value

LineSeries object.

Examples

```
LineSeries(name = "Sessions", data = c(150, 230, 224), smooth = TRUE)
```

LineStyle *Line Style*

Description

Stroke styling for lines in series, axes, and components.

Usage

```
LineStyle(
  color = NULL,
  width = NULL,
  opacity = NULL,
  type = NULL,
  cap = NULL,
  join = NULL,
  dash_offset = NULL,
  miter_limit = NULL,
  shadow_blur = NULL,
  shadow_color = NULL,
  shadow_offset_x = NULL,
  shadow_offset_y = NULL
)
```

Arguments

color	Optional Character: CSS line color.
width	Optional Numeric [0, Inf): Line width in pixels.
opacity	Optional Numeric [0, 1]: Line opacity.
type	Optional Character {"solid", "dashed", "dotted"}: Line type.
cap	Optional Character {"butt", "round", "square"}: Line cap.
join	Optional Character {"bevel", "round", "miter"}: Line join.
dash_offset	Optional Numeric: Dash offset for dashed lines.
miter_limit	Optional Numeric [0, Inf): Miter limit for miter joins.
shadow_blur	Optional Numeric [0, Inf): Shadow blur radius.
shadow_color	Optional Character: Shadow color.
shadow_offset_x	Optional Numeric: Shadow horizontal offset.
shadow_offset_y	Optional Numeric: Shadow vertical offset.

Details

Corresponds to LineStyleOption in src/util/types.ts (line 1154). ECharts docs: https://echarts.apache.org/en/option.html#series-line.lineStyle

Value

LineStyle object.

Examples

```
LineStyle(color = "#16a085", width = 2, type = "dashed")
```

MapLibreOption *MapLibre Render Option*

Description

The complete, validated render spec for a MapLibre choropleth: a MapModel (the data) plus all visual styling and an optional title. This is the MapLibre analog of EChartsOption / SigmaOption – the single object draw() dispatches on to emit a rtemis-map widget. Theme is *not* a property here; like every backend, theming is supplied to draw() and resolved uniformly.

Usage

```
MapLibreOption(
  model = NULL,
  classification = "quantile",
  colormap = "blues",
  num_classes = 5,
  opacity = 1,
  show_boundaries = TRUE,
  outline_width = 0.2,
  show_legend = TRUE,
  legend_position = "bottom-right",
  tooltip_position = "top-right",
  title = NULL
)
```

Arguments

`model`	MapModel or named list: The choropleth data (a list must contain a `rows` element).
`classification`	Character {"quantile", "equal", "jenks"}: Class-break method. `"quantile"` (equal counts), `"equal"` (equal intervals), or `"jenks"` (natural breaks).
`colormap`	Character: Color ramp. One of the sequential schemes `"blues"`, `"viridis"`, `"ylorrd"`, `"greens"`, `"magma"`, or the diverging schemes `"rdbu"`, `"rdylgn"`, `"spectral"`, `"brbg"`.
`num_classes`	Numeric [2, 12]: Number of color classes.
`opacity`	Numeric [0, 1]: Region fill opacity.
`show_boundaries`	Logical: Whether to draw region outlines.
`outline_width`	Numeric [0, Inf): Region outline width in pixels.
`show_legend`	Logical: Whether to show the legend.
`legend_position`	Character {"top-left", "top-right", "bottom-left", "bottom-right"}: Legend corner.
`tooltip_position`	Character {"top-left", "top-right", "bottom-left", "bottom-right"}: Hover tooltip corner.
`title`	Optional Character: Title (currently surfaced via the value label; reserved for future use).

Details

Most users never touch this directly – `draw_choropleth()` and `draw_map()` build it – but power users can construct it for full control: `draw(MapLibreOption(model = MapModel(...), colormap = "viridis"))`.

Its `to_list()` produces the `{ model, style, title }` payload consumed by the `rtemis-map` htmlwidget binding (the geometry is added by the `draw()` method, since it depends only on the model's resolution).

Value

MapLibreOption object.

Examples

```
model <- MapModel(
  rows = list(
    MapRow(location = "USA", value = 25.5),
    MapRow(location = "CAN", value = 12.1)
  )
)
MapLibreOption(model = model, classification = "quantile")
```

MapModel *Map Model*

Description

A complete, renderer-agnostic choropleth dataset: a list of MapRow objects, the administrative resolution that selects the geometry, the label for the value column, and the ordered tooltip field names. The R analog of the MapModel TypeScript interface in rtemislive (~/Code/live/src/lib/types.ts), consumed by the rtemis-map htmlwidget.

Usage

```
MapModel(
  rows = list(),
  resolution = "country",
  value_label = "value",
  tooltip_fields = character(0)
)
```

Arguments

rows	List: List of MapRow objects.
resolution	Character {"country", "state", "county"}: Administrative resolution. "country" joins on ISO-A3, "state" on 2-digit FIPS, "county" on 5-digit FIPS.
value_label	Character: Display label for the value column (labels the legend and tooltip value).
tooltip_fields	Character: Ordered names of the extra fields to show in the tooltip (keys into each MapRow's extras).

Details

Most users build this implicitly through draw_choropleth(); the constructor is exported for power users who assemble rows directly.

Value

MapModel object.

Examples

```
MapModel(
  rows = list(
    MapRow(location = "USA", value = 25.5),
    MapRow(location = "CAN", value = 12.1)
  )
)
```

MapRow *Map Row*

Description

One region's datum in a choropleth: a raw location key, the numeric value that colors the region, and optional extra fields to surface in the tooltip. The R analog of the MapRow TypeScript interface in rtemislive (~/Code/live/src/lib/types.ts).

Usage

```
MapRow(location = character(0), value = integer(0), extras = NULL)
```

Arguments

location	Character: Raw location key (FIPS / ISO / name) before normalization. Normalization to the canonical geometry id happens in the renderer.
value	Numeric: The value that colors the region.
extras	Optional named list: Extra column values to show in the tooltip, keyed by column name.

Value

MapRow object.

Examples

```
MapRow(location = "USA", value = 25.5)
```

MarkArea *Mark Area*

Description

Container for markArea bands attached to a series. data is a list where each element is a length-2 list of MarkAreaDataPoint (or plain named list) defining the two opposite corners of one band.

Usage

```
MarkArea(
  data = NULL,
  silent = NULL,
  precision = NULL,
  item_style = NULL,
  label = NULL,
  z2 = NULL
)
```

Arguments

data	Optional list: List of length-2 lists of MarkAreaDataPoint / plain named list. Each inner pair defines one band.
silent	Optional Logical: Whether hover / tooltip are disabled.
precision	Optional Integer [0, Inf): Decimal precision used by statistical methods ("min", "max", etc.).
item_style	Optional ItemStyle: Default fill/border styling for all bands. Per-band overrides go on the first MarkAreaDataPoint of a pair.
label	Optional LabelOption: Default label configuration.
z2	Optional Numeric: Front-back order among markers.

Details

Corresponds to MarkAreaOption in src/component/marker/MarkAreaModel.ts (line 60). ECharts docs: https://echarts.apache.org/en/option.html#series-line.markArea

Value

MarkArea object.

Examples

```
# One shaded band, defined by its two corner points.
MarkArea(
  data = list(list(
    MarkAreaDataPoint(x_axis = 1),
    MarkAreaDataPoint(x_axis = 3)
```

```
  ))
)
```

`MarkAreaDataPoint` *Mark Area Data Point*

Description

One endpoint of a `markArea` band. Two of these — wrapped in a length-2 list — define the opposite corners of a shaded rectangle. Supplying only `x_axis` on both endpoints produces a full-height vertical band (useful for highlighting x-ranges of a time series); supplying only `y_axis` produces a full-width horizontal band.

Usage

```
MarkAreaDataPoint(
  x_axis = NULL,
  y_axis = NULL,
  name = NULL,
  value = NULL,
  item_style = NULL,
  label = NULL
)
```

Arguments

`x_axis`	Optional Numeric or Character: Position on the x-axis.
`y_axis`	Optional Numeric or Character: Position on the y-axis.
`name`	Optional Character: Label text for the band.
`value`	Optional Numeric or Character: Fallback label value when `name` is not set.
`item_style`	Optional ItemStyle: Fill and border styling. Styling set on the first endpoint of a pair applies to the whole band.
`label`	Optional LabelOption: Label configuration.

Details

Corresponds to `MarkArea2DDataItemDimOption` in `src/component/marker/MarkAreaModel.ts` (line 49). ECharts docs: https://echarts.apache.org/en/option.html#series-line.markArea.data

Value

`MarkAreaDataPoint` object.

Examples

```
MarkAreaDataPoint(x_axis = 1, name = "Event window")
```

`MermaidConfig` *Mermaid Diagram Configuration*

Description

A serializable description of a Mermaid diagram rendered with Pele. Build one with `setup_MermaidConfig()` rather than calling this constructor directly.

Usage

```
MermaidConfig(
  dat_path = NULL,
  title = NULL,
  origin = NULL,
  writer = NULL,
  palette = NULL,
  node_color = NULL,
  fill_alpha = 0.1,
  font_size = 14,
  padding = NULL,
  auto_direction = FALSE,
  fit = FALSE
)
```

Arguments

`dat_path`	Optional Character: Path to the data, read at draw time. The serializable alternative to passing `data` to `draw()`.
`title`	Optional Character: Chart title. Declared here because it is the one property every chart type has; `palette` is not (four take a `colormap` instead), and axis labels and margins are cartesian-only.
`origin`	Optional Named character {"user", "default", "derived"}: Where each value came from, one entry per settable property. Absent on an authored config; written by the interface that resolved it.
`writer`	Optional Named character: Which interface wrote the config, as `name` and `version`. Absent on an authored config.
`palette`	Optional Character: Colors for categorical entities, overriding the theme palette. `NULL` uses the theme's.
`node_color`	Optional Named character: Colors for selected nodes, named by node id. Each named node gets a border in its color and a fill of the same color at `fill_alpha` opacity.
`fill_alpha`	Numeric [0, 1]: Fill opacity of the nodes in `node_color`.
`font_size`	Numeric (0, Inf): Label font size in pixels.
`padding`	Optional Numeric [0, Inf): Space around the diagram in pixels. `NULL` uses Pele's default.

auto_direction	Logical: Allow left-to-right flowcharts to be drawn top-to-bottom in widgets narrower than 640 pixels.
fit	Logical: Shrink the diagram to the widget width instead of drawing it at its natural size.

Details

The bound data is the Mermaid source: draw(config, data = source), where source is a character vector, or a .mmd or .mermaid file named by dat_path. See draw_mermaid() for the supported Mermaid syntax.

Value

MermaidConfig object.

Author(s)

EDG

Examples

```
setup_MermaidConfig(font_size = 16)@type
```

MinorSplitLine *Minor Split Line*

Description

Configuration for minor split lines (between main split lines).

Usage

```
MinorSplitLine(show = NULL, line_style = NULL)
```

Arguments

show	Optional Logical: Whether to show minor split lines.
line_style	Optional LineStyle: Line styling.

Details

Corresponds to MinorSplitLineOption in src/coord/axisCommonTypes.ts (line 364). ECharts docs: https://echarts.apache.org/en/option.html#xAxis.minorSplitLine

Value

MinorSplitLine object.

Examples

```
MinorSplitLine(show = TRUE, line_style = LineStyle(opacity = 0.2))
```

`MinorTick` *Minor Tick*

Description

Configuration for minor axis tick marks (between main ticks).

Usage

```
MinorTick(show = NULL, split_number = NULL, length = NULL, line_style = NULL)
```

Arguments

`show`	Optional Logical: Whether to show minor ticks.
`split_number`	Optional Numeric `[0, Inf)`: Number of minor ticks between two main ticks.
`length`	Optional Numeric `[0, Inf)`: Minor tick length in pixels.
`line_style`	Optional LineStyle: Line styling.

Details

Corresponds to `MinorTickOption` in `src/coord/axisCommonTypes.ts` (line 346). ECharts docs: https://echarts.apache.org/en/option.html#xAxis.minorTick

Value

`MinorTick` object.

Examples

```
MinorTick(show = TRUE, split_number = 5)
```

NetworkConfig *Network Graph Configuration*

Description

A serializable description of a network graph. Build one with `setup_NetworkConfig()` rather than calling this constructor directly.

Usage

```
NetworkConfig(
  dat_path = NULL,
  title = NULL,
  origin = NULL,
  writer = NULL,
  directed = FALSE,
  threshold = NULL,
  self_loops = FALSE,
  layout = "force",
  cluster_resolution = 1,
  scale_by_degree = TRUE,
  color_by_group = FALSE,
  blend_edges = FALSE,
  show_labels = TRUE,
  node_size = 10,
  edge_scale = 3,
  node_opacity = 0.95,
  edge_opacity = 0.4,
  palette = NULL,
  node_color = "#6CA3A0",
  positive_color = NULL,
  negative_color = NULL
)
```

Arguments

`dat_path`	Optional Character: Path to the data, read at draw time. The serializable alternative to passing `data` to `draw()`.
`title`	Optional Character: Chart title. Declared here because it is the one property every chart type has; `palette` is not (four take a `colormap` instead), and axis labels and margins are cartesian-only.
`origin`	Optional Named character {"user", "default", "derived"}: Where each value came from, one entry per settable property. Absent on an authored config; written by the interface that resolved it.
`writer`	Optional Named character: Which interface wrote the config, as `name` and `version`. Absent on an authored config.

`directed` Logical: Treat the matrix as directed.

`threshold` Optional Numeric: Drop edges weaker than this.

`self_loops` Logical: Keep self-edges.

`layout` Character: Layout algorithm.

`cluster_resolution`
Numeric `[0, Inf)`: Community-detection resolution. Higher values find more, smaller communities.

`scale_by_degree`
Logical: Size nodes by their degree.

`color_by_group` Logical: Color nodes by detected community.

`blend_edges` Logical: Blend overlapping edges.

`show_labels` Logical: Draw node labels.

`node_size` Numeric `[0, Inf)`: Base node size.

`edge_scale` Numeric `[0, Inf)`: Edge width multiplier.

`node_opacity`, `edge_opacity`
Numeric `[0, 1]`: Opacity of each.

`palette` Optional Character: Community colors, overriding the theme palette. `NULL` uses the theme's.

`node_color` Character: Node color when not colored by group.

`positive_color`, `negative_color`
Optional Character: Edge colors by sign. `NULL` uses the theme's sign colors.

Details

The bound data is an adjacency matrix, optionally with a node table: `draw(config, data = m)` or `draw(config, data = list(adjacency = m, nodes = df))`.

Value

`NetworkConfig` object.

Author(s)

EDG

Examples

```
setup_NetworkConfig(layout = "force")@type
```

PanelLayout *Rectangular Panel Layout*

Description

Package-owned, backend-independent layout settings. Child charts retain separate ECharts options (`EChartsOption` in `src/export/option.ts`); this layout does not merge or renumber their axes, series, or legends. Rows fill from left to right, with equally sized cells. Unused final cells remain empty. Settings serialize as ordinary JSON with `to_list()`.

Usage

```
PanelLayout(ncol = 1L, gap = 12, padding = 0)
```

Arguments

`ncol`	Integer `[1, Inf)`: Number of columns.
`gap`	Numeric `[0, Inf)`: Finite gap between cells in pixels.
`padding`	Numeric `[0, Inf)`: Finite outer padding in pixels.

Value

PanelLayout: Validated layout settings.

Examples

```
to_list(setup_PanelLayout(ncol = 2))
```

panel_layout_schema *JSON Schema for Panel Layout Settings*

Description

Uses the S7 properties' declarations and omits defaults. This is a layout schema, separate from the chart registry and child chart/data contracts. Resolved layouts emitted by `to_list()` contain every setting.

Usage

```
panel_layout_schema(complete = FALSE)
```

Arguments

`complete`	Logical: Require every setting instead of allowing omissions.

Value

Named list: Draft 2020-12 JSON Schema.

Examples

```
panel_layout_schema()[["properties"]][["ncol"]]
```

`PeleOption` *Pele Render Option*

Description

The complete, validated render specification for a Mermaid diagram drawn with Pele: the Mermaid source plus its render settings. This is the Pele analog of EChartsOption, SigmaOption, and MapLibreOption, the object `draw()` dispatches on to create a `rtemis-diagram` widget. The theme is an argument to `draw()`, not a property.

Usage

```
PeleOption(
  source = character(0),
  font_size = 14,
  padding = NULL,
  auto_direction = FALSE,
  fit = FALSE,
  palette = NULL,
  node_color = NULL,
  fill_alpha = 0.1,
  critical_color = "#EA384A"
)
```

Arguments

`source`	Character: Mermaid diagram text, one string.
`font_size`	Numeric (0, Inf): Label font size in pixels.
`padding`	Optional Numeric [0, Inf): Space around the diagram in pixels. `NULL` uses Pele's default.
`auto_direction`	Logical: Allow Pele to draw left-to-right flowcharts top-to-bottom in elements narrower than 640 pixels.
`fit`	Logical: Shrink the diagram to the widget width, with pinch and Ctrl/Cmd-wheel zoom. `FALSE` draws it at its natural size in a scrolling widget.
`palette`	Optional Character: Colors for categorical entities, such as pie slices and mindmap branches. Pele uses the first eight. `NULL` uses the theme palette.
`node_color`	Optional Named character: Colors for selected nodes, named by node id. Each named node gets a border in its color and a fill of the same color at `fill_alpha` opacity.

`fill_alpha`	Numeric [0, 1]: Fill opacity of the nodes in `node_color`.
`critical_color`	Character: Color of critical Gantt tasks and high-priority kanban items.

Details

Most users call `draw_mermaid()`, which builds this object.

Value

`PeleOption` object.

Author(s)

EDG

Examples

```
PeleOption(source = "flowchart LR\n  A --> B")
```

PieConfig *Pie Chart Configuration*

Description

A serializable description of a pie chart. Build one with `setup_PieConfig()` rather than calling this constructor directly.

Usage

```
PieConfig(
  dat_path = NULL,
  title = NULL,
  origin = NULL,
  writer = NULL,
  legend_position = "top",
  legend_placement = "outside",
  legend_title = NULL,
  values = NULL,
  labels = NULL,
  label_format = "name",
  percent_digits = 1L,
  rose_type = NULL,
  radius = "75%",
  palette = NULL
)
```

Arguments

dat_path	Optional Character: Path to the data, read at draw time. The serializable alternative to passing data to draw().
title	Optional Character: Chart title. Declared here because it is the one property every chart type has; palette is not (four take a colormap instead), and axis labels and margins are cartesian-only.
origin	Optional Named character {"user", "default", "derived"}: Where each value came from, one entry per settable property. Absent on an authored config; written by the interface that resolved it.
writer	Optional Named character: Which interface wrote the config, as name and version. Absent on an authored config.
legend_position	Character {"top", "bottom", "left", "right", "top-left", "top-right", "bottom-left", "bottom-right"}: Legend anchor. Top/bottom anchors use horizontal rows; left/right anchors use a vertical column. Corner anchors align within the top or bottom row.
legend_placement	Character {"outside", "inside"}: Relation to the plotting area. Outside placement reserves space for the complete legend; inside placement overlays the data. Neither setting adds a missing legend.
legend_title	Optional Character: Title drawn with the legend or colorbar: above left/right legends and vertical colorbars, before the first row of top/bottom legends and horizontal colorbars. NULL draws no title. A title does not add a missing legend.
values	Optional Character: Column holding the slice values.
labels	Optional Character: Column holding the slice labels.
label_format	Character {"name", "value", "percent", "name_value", "name_percent"}: Slice label content.
percent_digits	Integer [0, 6]: Percentage precision.
rose_type	Optional Character {"radius", "area"}: Draw as a Nightingale rose chart, sizing slices by radius or by area. NULL draws a plain pie.
radius	Character: Outer radius, as a CSS length or percentage.
palette	Optional Character: Slice colors, overriding the theme palette for this chart. NULL uses the theme's.

Value

PieConfig object.

Author(s)

EDG

Examples

```
setup_PieConfig(values = "n", labels = "kind")@type
```

PieLabelOption *Pie Label Options*

Description

Outside-label alignment for pie and rose charts. Extends LabelOption. Corresponds to `PieLabelOption` in `src/chart/pie/PieSeries.ts`. ECharts docs: https://echarts.apache.org/en/option.html#series-pie.label

Usage

```
PieLabelOption(
  show = NULL,
  position = NULL,
  distance = NULL,
  rotate = NULL,
  offset = NULL,
  formatter = NULL,
  silent = NULL,
  precision = NULL,
  value_animation = NULL,
  min_margin = NULL,
  text_style = NULL,
  align_to = NULL,
  edge_distance = NULL,
  bleed_margin = NULL,
  distance_to_label_line = NULL
)
```

Arguments

`show`	Optional Logical: Whether to show the label.
`position`	Optional Character {"top", "left", "right", "bottom", "inside", "insideLeft", "insideRight", "insideTop", "insideBottom", "insideTopLeft", "insideBottomLeft", "insideTopRight", "insideBottomRight", "outside"} or Numeric: Label position. `"bottom"`, `"inside"`, `"insideLeft"`, `"insideRight"`, `"insideTop"`, `"insideBottom"`, `"insideTopLeft"`, `"insideBottomLeft"`, `"insideTopRight"`, `"insideBottomRight"`, `"outside"`.
`distance`	Optional Numeric: Distance from the element.
`rotate`	Optional Numeric: Rotation angle in degrees.
`offset`	Optional Numeric: Offset `c(x, y)` in pixels.
`formatter`	Optional Character or function: Label text formatter.
`silent`	Optional Logical: Whether the label is silent.
`precision`	Optional Numeric or Character {"auto"}: Numeric precision.

value_animation
: Optional Logical: Whether to animate value changes.

min_margin
: Optional Numeric `[0, Inf)`: Minimum margin between labels.

text_style
: Optional TextStyle: Text appearance.

align_to
: Optional Character {"none", "labelLine", "edge"}: Outside-label alignment.

edge_distance
: Optional Numeric or Character: Distance from the plot edge.

bleed_margin
: Optional Numeric: Space outside labels when not edge-aligned.

distance_to_label_line
: Optional Numeric: Gap between label and guide line.

Value

A `PieLabelOption` object.

Examples

```
PieLabelOption(align_to = "edge", edge_distance = 8)
```

PieSeries *Pie Series*

Description

Configuration for a pie chart series.

Usage

```
PieSeries(
  name = NULL,
  data = NULL,
  silent = NULL,
  legend_hover_link = NULL,
  color = NULL,
  z_level = NULL,
  z = NULL,
  center = NULL,
  radius = NULL,
  rose_type = NULL,
  clockwise = NULL,
  start_angle = NULL,
  end_angle = NULL,
  pad_angle = NULL,
  min_angle = NULL,
  min_show_label_angle = NULL,
  selected_offset = NULL,
  avoid_label_overlap = NULL,
```

```
  percent_precision = NULL,
  still_show_zero_sum = NULL,
  animation_type = NULL,
  item_style = NULL,
  label = NULL,
  label_line = NULL,
  left = NULL,
  right = NULL,
  top = NULL,
  bottom = NULL
)
```

Arguments

`name`	Optional Character: Series name.
`data`	Optional Numeric or list: Pie data. A list of named lists with `value` and `name` fields, or a numeric vector.
`silent`	Optional Logical: Whether the series is silent.
`legend_hover_link`	Optional Logical: Whether to highlight related legends on hover.
`color`	Optional Character: Series color override.
`z_level`	Optional Numeric: Canvas layer index.
`z`	Optional Numeric: Front-back order within the same layer.
`center`	Optional Numeric or Character: Pie center as a length-2 vector.
`radius`	Optional Numeric or Character: Pie radius.
`rose_type`	Optional Character {"radius", "area"}: Nightingale chart type.
`clockwise`	Optional Logical: Whether to lay out sectors clockwise.
`start_angle`	Optional Numeric: Starting angle in degrees.
`end_angle`	Optional Numeric or Character {"auto"}: Ending angle in degrees.
`pad_angle`	Optional Numeric `[0, Inf)`: Padding angle between sectors in degrees.
`min_angle`	Optional Numeric `[0, Inf)`: Minimum sector angle.
`min_show_label_angle`	Optional Numeric `[0, Inf)`: Minimum angle required to show labels.
`selected_offset`	Optional Numeric `[0, Inf)`: Offset of a selected sector.
`avoid_label_overlap`	Optional Logical: Whether to avoid label overlap automatically.
`percent_precision`	Optional Numeric `[0, Inf)`: Decimal precision for percentages.
`still_show_zero_sum`	Optional Logical: Whether to show the pie when all values are zero.
`animation_type`	Optional Character {"expansion", "scale"}: First-render animation type.
`item_style`	Optional ItemStyle: Pie-sector styling.

label	Optional LabelOption: Data-label configuration.
label_line	Optional LabelLine: Label connector line configuration.
left	Optional Numeric or Character: Left position.
right	Optional Numeric or Character: Right position.
top	Optional Numeric or Character: Top position.
bottom	Optional Numeric or Character: Bottom position.

Details

Corresponds to PieSeriesOption in src/chart/pie/PieSeries.ts (line 105). ECharts docs: https://echarts.apache.org/en/option.html#series-pie

Value

PieSeries object.

Examples

```
PieSeries(
  name = "Browser",
  data = list(
    list(value = 1048, name = "Chrome"),
    list(value = 735, name = "Firefox")
  ),
  radius = "50%"
)
```

---

read_chart_config *Read a chart config from JSON*

Description

Reconstructs through the chart's setup_* function rather than its constructor, so a document from any source – another interface, a hand-written file, a language model – arrives resolved through the same seam a direct call goes through.

Usage

```
read_chart_config(x)
```

Arguments

x	Character: Path to a JSON file, or a JSON string.

Value

ChartConfig subclass object.

Author(s)

EDG

Examples

```
path <- tempfile(fileext = ".json")
write_chart_config(setup_ScatterConfig(x = "wt", y = "mpg", fit = "glm"), path)
read_chart_config(path)@fit
unlink(path)
```

---

renderDraw *Shiny render function for draw widget*

Description

Shiny render function for draw widget

Usage

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

Arguments

expr	Expression: Expression that returns a draw widget.
env	Environment: Evaluation environment.
quoted	Logical: Whether expr is quoted.

Value

Shiny render function for a draw widget.

Examples

```
if (requireNamespace("shiny", quietly = TRUE)) {
  renderDraw(draw_bar(x = c("A", "B", "C"), y = c(3, 7, 2)))
}
```

`resolve` *Resolve a partial chart config into a complete one*

Description

The middle step of the pipeline: **partial config -> complete config -> render**. Fills in every value that can be derived from the data, and stamps each one `"derived"` in the config's `origin` map.

Usage

```
resolve(config, data = NULL, ...)
```

Arguments

`config`	ChartConfig: The chart configuration.
`data`	Optional Data frame or named list: The data to plot. When `NULL`, the config's `dat_path` is read if it has one.
`...`	Passed to methods.

Details

Resolution is its own step rather than a set of fallbacks inside the option builders, and that is what makes an output config honest. A value derived during rendering would have to be *reconstructed* to write it down; a value derived here is simply part of the document that then gets rendered. Reading a complete config and rendering it become the same operation.

Two things are deliberately **not** resolved:

- **Anything describing the display surface** – width, height, aspect ratio, container geometry. Those belong to the interface, are recomputed by each one, and are never written into a document: a height computed for an IDE pane is wrong in a large web canvas.
- **Anything with nothing to derive from.** Labels come from column names, so a config that names no columns gets no labels.

Resolving is **idempotent** and never overwrites a value that is already set, so resolving twice is the same as resolving once. `compile()` relies on that: it resolves up front, and a builder it shares with `draw()` may resolve again without changing the result.

Data is optional. With none available, the values that need it are simply not derived; the ones that come from the column names still are. That makes `resolve()` total – it has no failing input – so a caller can always ask for the most complete config obtainable from what it has.

Value

ChartConfig subclass object, with every derivable value filled in.

Author(s)

EDG

Examples

```
# Axis limits come from the data; the labels come from the column names.
resolve(setup_ScatterConfig(x = "wt", y = "mpg"), data = mtcars)@xlim
```

ROCConfig *ROC Plot Configuration*

Description

Portable bindings for long ROC records. Each split/class/fold identifies one curve. The reserved fold label aggregate identifies pooled predictions. Compiles to LineSeriesOption in src/chart/line/LineSeries.ts, GridOption in src/coord/cartesian/GridModel.ts, and LegendOption in src/component/legend/LegendModel.ts.

Usage

```
ROCConfig(
  dat_path = NULL,
  title = NULL,
  origin = NULL,
  writer = NULL,
  legend_position = "top",
  legend_placement = "outside",
  legend_title = NULL,
  fpr = "fpr",
  tpr = "tpr",
  auc = NULL,
  class_label = NULL,
  split = NULL,
  fold = NULL,
  omitted = NULL,
  variant = "aggregate",
  digits = 3L,
  diagonal = TRUE,
  diagonal_color = "#888888",
  palette = NULL,
  legend = TRUE,
  square = TRUE,
  line_width = 2,
  fold_opacity = 0.45,
  xlab = "False positive rate",
  ylab = "True positive rate"
)
```

Arguments

dat_path	Optional Character: Path to the data, read at draw time. The serializable alternative to passing data to draw().

title	Optional Character: Chart title.
origin	Optional Named character {"user", "default", "derived"}: Where each value came from, one entry per settable property. Absent on an authored config; written by the interface that resolved it.
writer	Optional Named character: Which interface wrote the config, as name and version. Absent on an authored config.
legend_position	Character {"top", "bottom", "left", "right", "top-left", "top-right", "bottom-left", "bottom-right"}: Legend anchor. Top/bottom anchors use horizontal rows; left/right anchors use a vertical column. Corner anchors align within the top or bottom row.
legend_placement	Character {"outside", "inside"}: Relation to the plotting area. Outside placement reserves space for the complete legend; inside placement overlays the data. Neither setting adds a missing legend.
legend_title	Optional Character: Title drawn with the legend or colorbar: above left/right legends and vertical colorbars, before the first row of top/bottom legends and horizontal colorbars. NULL draws no title. A title does not add a missing legend.
fpr	Character: Column containing false positive rates.
tpr	Character: Column containing true positive rates.
auc	Optional Character: Optional supplied AUC column; unset integrates the supplied vertices.
class_label	Optional Character: Optional positive-class label column.
split	Optional Character: Optional sample or model label column.
fold	Optional Character: Optional resample column; aggregate identifies pooled curves.
omitted	Optional Character: Optional omitted-observation count column, constant within each curve.
variant	Character: Draw pooled curves or separate resample curves.
digits	Integer: Decimal places for AUC labels and tooltip values.
diagonal	Logical: Show an independent chance diagonal.
diagonal_color	Character: Chance-line color.
palette	Optional Character: Group colors; unset uses the chart theme.
legend	Logical: Show group labels and AUC summaries.
square	Logical: Keep the plotting grid square.
line_width	Numeric: Curve stroke width in pixels.
fold_opacity	Numeric: Opacity of individual resample curves.
xlab	Character: Horizontal axis label.
ylab	Character: Vertical axis label.

Value

A ROCConfig object.

Statistical semantics

Vertices are ordered by FPR then TPR and joined without smoothing. Curves must be monotone and include (0, 0) and (1, 1). Supplied AUC is authoritative and may describe a higher-resolution curve. Without an AUC binding, the supplied vertices define trapezoidal AUC. An undefined curve is represented by missing FPR, TPR, and AUC and is reported in the console when omitted. Per-resample legends report the unweighted mean and sample SD of defined fold AUCs, with available/total curve counts. SD is NA for one defined fold. Pooled AUC is never inferred from per-fold AUCs. The legend uses the shared position and placement controls. Curve identity and AUC share a line; long labels wrap to the available width. All entries remain present in static SVG output, without a scrolling legend. Hover describes the selected curve, including its resample identifier. Numeric tooltip labels use `digits`; coordinates and AUCs retain their full precision in the compiled data.

Examples

```
draw(setup_ROCConfig(), data = data.frame(fpr = c(0, 0, 1), tpr = c(0, 1, 1)))
```

`roc_probabilities` *Normalize labels and probability columns by class identity*

Description

Normalize labels and probability columns by class identity

Usage

```
roc_probabilities(x, predicted_prob, positive = NULL)
```

Arguments

`x`	Factor or atomic vector: Reference labels.
`predicted_prob`	Numeric vector or matrix: Class probabilities.
`positive`	Optional Character: Binary positive class.

Value

List containing reference labels, class levels, selected classes, and scores.

Examples

```
roc_probabilities(factor(c("no", "yes")), c(0.1, 0.9))
```

roc_vertices *Compute empirical ROC vertices with whole tied-score blocks*

Description

Higher scores always indicate the named positive class. Consuming a tied block at once yields trapezoidal AUC equivalent to half credit for ties. The direction is never selected from the observed performance.

Usage

```
roc_vertices(x)
```

Arguments

x	List: Normalized labels and probabilities from roc_probabilities().

Value

Data frame of class, FPR, TPR, AUC, and omitted-pair counts.

Examples

```
roc_vertices(roc_probabilities(factor(c("no", "yes")), c(0.1, 0.9)))
```

SankeyConfig *Sankey Diagram Configuration*

Description

A serializable description of a Sankey diagram. Build one with `setup_SankeyConfig()` rather than calling this constructor directly.

Usage

```
SankeyConfig(
  dat_path = NULL,
  title = NULL,
  origin = NULL,
  writer = NULL,
  source = "source",
  target = "target",
  value = "value",
  orient = "horizontal",
  node_align = NULL,
  node_width = NULL,
```

```
  node_gap = NULL,
  palette = NULL
)
```

Arguments

dat_path	Optional Character: Path to the data, read at draw time. The serializable alternative to passing `data` to `draw()`.
title	Optional Character: Chart title. Declared here because it is the one property every chart type has; `palette` is not (four take a `colormap` instead), and axis labels and margins are cartesian-only.
origin	Optional Named character {"user", "default", "derived"}: Where each value came from, one entry per settable property. Absent on an authored config; written by the interface that resolved it.
writer	Optional Named character: Which interface wrote the config, as `name` and `version`. Absent on an authored config.
source	Character: Column naming each link's source node.
target	Character: Column naming each link's target node.
value	Character: Column holding each link's magnitude.
orient	Character {"horizontal", "vertical"}: Layout direction.
node_align	Optional Character {"left", "right", "justify"}: How nodes are aligned along the flow.
node_width	Optional Numeric `[0, Inf)`: Node thickness in pixels.
node_gap	Optional Numeric `[0, Inf)`: Gap between nodes in pixels.
palette	Optional Character: Node colors, overriding the theme palette for this chart. `NULL` uses the theme's.

Details

The bound data is a table of links, one row per flow. `source`, `target` and `value` name its columns.

Value

`SankeyConfig` object.

Author(s)

EDG

Examples

```
setup_SankeyConfig()@type
```

SankeyEdgeItem *Sankey Edge Item*

Description

One directed link (edge) in a Sankey diagram.

Usage

```
SankeyEdgeItem(
  source = NULL,
  target = NULL,
  value = NULL,
  line_style = NULL,
  edge_label = NULL,
  focus_node_adjacency = NULL
)
```

Arguments

source	Optional Character: Source node name.
target	Optional Character: Target node name.
value	Optional Numeric: Flow value (determines link width).
line_style	Optional LineStyle or list: Link line styling. Accepts a LineStyle object or a plain named list; a list also allows the Sankey-specific curveness field (e.g. list(curveness = 0.5, opacity = 0.3)).
edge_label	Optional LabelOption: Label shown on the link itself.
focus_node_adjacency	Optional Logical or Character {"inEdges", "outEdges", "allEdges"}: Which adjacent elements to highlight on hover.

Details

Corresponds to SankeyEdgeItemOption in src/chart/sankey/SankeySeries.ts (line 88). ECharts docs: https://echarts.apache.org/en/option.html#series-sankey.links

Value

SankeyEdgeItem object.

Examples

```
SankeyEdgeItem(source = "A", target = "B", value = 5)
```

SankeyLevelOption *Sankey Level Option*

Description

Style overrides applied to all nodes at a given column depth. `depth = 0` targets the leftmost column.

Usage

```
SankeyLevelOption(
  depth = integer(0),
  label = NULL,
  item_style = NULL,
  line_style = NULL
)
```

Arguments

`depth`	Numeric `[0, Inf)`: Column depth to target (0 = leftmost). Required.
`label`	Optional LabelOption: Node label configuration at this depth.
`item_style`	Optional ItemStyle: Node fill/border styling at this depth.
`line_style`	Optional LineStyle or list: Link styling for edges leaving this depth. Accepts a LineStyle object or a plain list (allows `curveness`).

Details

Corresponds to `SankeyLevelOption` in `src/chart/sankey/SankeySeries.ts` (line 96). ECharts docs: https://echarts.apache.org/en/option.html#series-sankey.levels

Value

`SankeyLevelOption` object.

Examples

```
SankeyLevelOption(depth = 0L, item_style = ItemStyle(color = "#16a085"))
```

SankeyNodeItem *Sankey Node Item*

Description

One node in a Sankey diagram. Node names must match the `source` / `target` strings used in the corresponding SankeyEdgeItem objects.

Usage

```
SankeyNodeItem(
  name = NULL,
  value = NULL,
  depth = NULL,
  draggable = NULL,
  local_x = NULL,
  local_y = NULL,
  label = NULL,
  item_style = NULL,
  focus_node_adjacency = NULL
)
```

Arguments

`name`	Optional Character: Node name (must match source/target strings).
`value`	Optional Numeric: Node value shown in tooltip.
`depth`	Optional Numeric `[0, Inf)`: Fixed column depth (0 = leftmost).
`draggable`	Optional Logical: Whether the node can be dragged.
`local_x`	Optional Numeric: Horizontal position inside the layout rect.
`local_y`	Optional Numeric: Vertical position inside the layout rect.
`label`	Optional LabelOption: Node label configuration.
`item_style`	Optional ItemStyle: Node fill/border styling.
`focus_node_adjacency`	Optional Logical or Character {"inEdges", "outEdges", "allEdges"}: Which adjacent elements to highlight on hover.

Details

Corresponds to `SankeyNodeItemOption` in `src/chart/sankey/SankeySeries.ts` (line 73). ECharts docs: https://echarts.apache.org/en/option.html#series-sankey.data

Value

`SankeyNodeItem` object.

Examples

```
SankeyNodeItem(name = "Source A", value = 10)
```

`SankeySeries` *Sankey Series*

Description

Configuration for a Sankey diagram series. Nodes and directed links are represented as two separate lists. Node names in `data` must match the `source` and `target` strings in `links`.

Usage

```
SankeySeries(
  name = NULL,
  data = NULL,
  links = NULL,
  levels = NULL,
  orient = NULL,
  node_width = NULL,
  node_gap = NULL,
  node_align = NULL,
  draggable = NULL,
  layout_iterations = NULL,
  sort = NULL,
  roam = NULL,
  label = NULL,
  edge_label = NULL,
  line_style = NULL,
  item_style = NULL,
  color = NULL,
  left = NULL,
  right = NULL,
  top = NULL,
  bottom = NULL,
  silent = NULL,
  z_level = NULL,
  z = NULL
)
```

Arguments

`name`	Optional Character: Series name for legend and tooltip.
`data`	Optional list: Nodes as a list of SankeyNodeItem objects or plain named lists, each with at least a `name` field.

links	Optional list: Edges as a list of SankeyEdgeItem objects or plain named lists with source, target, and value fields.
levels	Optional list: List of SankeyLevelOption for per-depth styling.
orient	Optional Character {"horizontal", "vertical"}: Flow direction.
node_width	Optional Numeric [0, Inf): Width of each node rectangle in pixels.
node_gap	Optional Numeric [0, Inf): Vertical gap between nodes in pixels.
node_align	Optional Character {"justify", "left", "right"}: Node alignment within columns.
draggable	Optional Logical: Whether nodes can be dragged interactively.
layout_iterations	Optional Numeric [0, Inf): Number of layout iterations.
sort	Optional Character {"desc"} or NULL: Column-internal node ordering. NULL preserves the original data order.
roam	Optional Logical or Character {"move", "scale"}: Whether to enable pan/zoom.
label	Optional LabelOption: Node label configuration.
edge_label	Optional LabelOption: Edge label configuration.
line_style	Optional LineStyle or list: Default link line styling. Accepts a LineStyle object or a plain list; a plain list also allows the Sankey-specific curveness field (e.g. list(curveness = 0.5)).
item_style	Optional ItemStyle: Default node styling.
color	Optional Character: Color palette for nodes as a character vector.
left	Optional Numeric or Character: Series left offset.
right	Optional Numeric or Character: Series right offset.
top	Optional Numeric or Character: Series top offset.
bottom	Optional Numeric or Character: Series bottom offset.
silent	Optional Logical: Whether to disable mouse/touch events.
z_level	Optional Numeric: Canvas layer index.
z	Optional Numeric: Front-back order within the same canvas layer.

Details

Corresponds to SankeySeriesOption in src/chart/sankey/SankeySeries.ts (line 100). ECharts docs: https://echarts.apache.org/en/option.html#series-sankey

Value

SankeySeries object.

Examples

```
SankeySeries(
  data = list(SankeyNodeItem(name = "A"), SankeyNodeItem(name = "B")),
  links = list(SankeyEdgeItem(source = "A", target = "B", value = 5))
)
```

save_drawing *Save a Draw Widget to a File*

Description

Exports a widget created by draw() (or any of the draw_* functions) to a static file. Supports .svg for ECharts, Sigma networks, MapLibre maps, and Pele diagrams via Node.js server-side rendering, including complete draw_panels() figures. Requires a Node.js (>= 18) node binary on PATH.

Usage

```
save_drawing(widget, filename, width = NULL, height = NULL)
```

Arguments

widget	htmlwidget: A widget returned by draw() or a draw_* function.
filename	Character scalar: Nonempty output path with an extension.
width	Optional Numeric scalar (0, Inf): Finite image width in pixels. Unset uses the widget's numeric width, or 800 for a relative/unspecified width.
height	Optional Numeric scalar (0, Inf): Finite image height in pixels. Unset uses the widget's numeric height, or 600 for a relative/unspecified height.

Details

Zoom controls (sliders and inside zoom) and toolbox buttons are omitted, and the band reserved for a zoom slider returns to the plot; the export shows the full data range. Callbacks under tooltips and axis pointers are omitted. Other JavaScript callbacks are rejected because discarding them could change visible content. Built-in Gantt, dendrogram, and boxplot point renderers are shared with the browser widget and use JSON parameters. Networks share their deterministic layout and styles with the interactive widget. Maps share geometry, location joins, and color scales and use a Mercator projection. Exports show the initial configured view; browser camera changes and hover states are not stored in the R widget. SVG text remains editable. Network labels are fitted to the page and suppressed when they overlap. Diagrams export at their natural size, so width and height do not apply to them; their colors are written as literal values. Diagram labels are measured in the first installed font of the theme's font stack, read with the 'systemfonts' package, and the file names that font followed by Helvetica. When the stack reaches a generic family such as system-ui first, or 'systemfonts' is not installed, labels are measured and written in Helvetica or a font with the same metrics.

Value

The filename, invisibly.

Examples

```
chart <- draw_bar(x = c("A", "B", "C"), y = c(3, 7, 2))
path <- file.path(tempdir(), "chart.svg")
# SVG export shells out to Node.js; report instead of failing without it.
tryCatch(
  save_drawing(chart, path),
  rtemis_export_error = function(e) message(conditionMessage(e))
)
unlink(path)
```

Scatter3DConfig *Three-Dimensional Scatter Configuration*

Description

Uses ECharts-GL Scatter3DSeries, Grid3DModel, Cartesian3D, and OrbitControl. Coordinates are independently scaled to equal box dimensions. Perspective projection scales coordinates by camera depth; orthographic projection keeps parallel lines parallel. Exported point positions match the configured browser view. Browser rotation/zoom is interactive; saving the R widget exports its configured initial camera.

Usage

```
Scatter3DConfig(
  dat_path = NULL,
  title = NULL,
  origin = NULL,
  writer = NULL,
  x = NULL,
  y = NULL,
  z = NULL,
  group = NULL,
  alpha = 20,
  beta = 40,
  projection = "perspective",
  distance = 250,
  view_size = 190,
  point_size = 8,
  opacity = NULL,
  mode = "points",
  order = "input",
  line_width = 2,
  xlab = NULL,
  ylab = NULL,
  zlab = NULL,
  palette = NULL,
  fit = NULL,
```

```
  fit_params = NULL,
  n_fit = 25L,
  fit_alpha = 0.35
)
```

Arguments

dat_path	Optional Character: Path to the data, read at draw time. The serializable alternative to passing data to draw().
title	Optional Character: Chart title. Declared here because it is the one property every chart type has; palette is not (four take a colormap instead), and axis labels and margins are cartesian-only.
origin	Optional Named character {"user", "default", "derived"}: Where each value came from, one entry per settable property. Absent on an authored config; written by the interface that resolved it.
writer	Optional Named character: Which interface wrote the config, as name and version. Absent on an authored config.
x, y, z	Optional Character: Numeric coordinate column bindings.
group	Optional Character: Group column binding.
alpha	Numeric [-80, 80]: Camera elevation in degrees.
beta	Numeric [-180, 180]: Camera azimuth in degrees.
projection	Character {"perspective", "orthographic"}: Camera projection.
distance	Numeric [100, 400]: Perspective camera distance from the center of the 100-unit cube. Narrow and portrait layouts increase the distance to preserve space for axes.
view_size	Numeric [175, 400]: Orthographic viewing extent for a 100-unit cube.
point_size	Numeric (0, 100]: Marker diameter in pixels.
opacity	Optional Numeric [0, 1]: Point and line opacity. NULL sets it from the number of complete observations, as in draw_scatter()'s point_alpha, and draws paths alone (mode = "lines") opaque.
mode	Character c("points", "lines", "both"): Geometry to draw.
order	Character c("input", "x"): Order within complete path runs. Missing coordinates break paths; sorting never joins across those breaks.
line_width	Numeric (0, 100]: Path width in pixels.
xlab, ylab, zlab	Optional Character: Axis names.
palette	Optional Character: Group colors.
fit	Optional Character: Model of z on x and y drawn as a surface per group: "glm", "gam", or an rtemis supervised learning algorithm name such as "LINAD", which requires the rtemis package (>= 1.4.1). See draw_scatter(). NULL draws no surface.
fit_params	Optional Named list: Arguments passed to the learner named in fit. See draw_scatter().
n_fit	Integer [2, Inf): Grid points per axis for fitted surfaces.
fit_alpha	Numeric [0, 1]: Opacity of fitted surfaces.

Value

Scatter3DConfig: Validated portable settings.

Examples

```
setup_Scatter3DConfig(x = "Sepal.Length", y = "Sepal.Width", z = "Petal.Length")
```

ScatterConfig *Scatter Chart Configuration*

Description

A serializable description of a scatter chart: which columns it binds, its semantics, and its appearance. Build one with `setup_ScatterConfig()` rather than calling this constructor directly.

Usage

```
ScatterConfig(
  dat_path = NULL,
  title = NULL,
  origin = NULL,
  writer = NULL,
  legend_position = "top",
  legend_placement = "outside",
  legend_title = NULL,
  x = NULL,
  y = NULL,
  size = NULL,
  group = NULL,
  hover = NULL,
  fit_name = NULL,
  rug = FALSE,
  fit = NULL,
  fit_params = NULL,
  se = TRUE,
  se_times = 1.96,
  rsq = FALSE,
  diagonal = FALSE,
  diagonal_color = NULL,
  n_fit = 200L,
  zoom = FALSE,
  fit_alpha = 0.25,
  point_alpha = NULL,
  palette = NULL,
  square = FALSE,
  equal_axes = FALSE,
  pad = 0.04,
```

```
  xlim = NULL,
  ylim = NULL,
  xlab = NULL,
  ylab = NULL,
  margin_top = NULL,
  margin_right = NULL,
  margin_bottom = NULL,
  margin_left = NULL
)
```

Arguments

dat_path	Optional Character: Path to the data, read at draw time. The serializable alternative to passing data to draw().
title	Optional Character: Chart title. Declared here because it is the one property every chart type has; palette is not (four take a colormap instead), and axis labels and margins are cartesian-only.
origin	Optional Named character {"user", "default", "derived"}: Where each value came from, one entry per settable property. Absent on an authored config; written by the interface that resolved it.
writer	Optional Named character: Which interface wrote the config, as name and version. Absent on an authored config.
legend_position	Character {"top", "bottom", "left", "right", "top-left", "top-right", "bottom-left", "bottom-right"}: Legend anchor. Top/bottom anchors use horizontal rows; left/right anchors use a vertical column. Corner anchors align within the top or bottom row.
legend_placement	Character {"outside", "inside"}: Relation to the plotting area. Outside placement reserves space for the complete legend; inside placement overlays the data. Neither setting adds a missing legend.
legend_title	Optional Character: Title drawn with the legend or colorbar: above left/right legends and vertical colorbars, before the first row of top/bottom legends and horizontal colorbars. NULL draws no title. A title does not add a missing legend.
x, y	Optional Character: Columns drawn on each axis.
size	Optional Character: Column giving per-point size.
group	Optional Character: Column to group and color points by.
hover	Optional Character: Column containing per-observation tooltip labels.
fit_name	Optional Character: Label for fitted layers.
rug	Logical: Show marginal marks along the x and y axes.
fit	Optional Character: Fit to overlay: "glm", "gam", or an rtemis supervised learning algorithm name such as "LINAD", which requires the rtemis package. NULL draws no fit. See draw_scatter().
fit_params	Optional Named list: Arguments passed to the learner named in fit, e.g. list(max_leaves = 8) for "LINAD". See draw_scatter().

se	Logical: If TRUE, shade the fit standard-error band.
se_times	Numeric [0, Inf): Multiplier for the fitted standard error.
rsq	Logical: Include the fitted model's R-squared in series labels.
diagonal	Logical: Draw the identity line within the axis limits.
diagonal_color	Optional Character: Identity-line color.
n_fit	Integer [2, Inf): Points used to draw the fit line.
zoom	Logical: Enable Shift+wheel zoom, drag panning, and box-zoom controls on both axes. With square or equal_axes, the controls omit box zoom, which sets each axis independently.
fit_alpha	Numeric [0, 1]: Opacity of the standard-error band.
point_alpha	Optional Numeric [0, 1]: Point opacity. NULL sets it from the number of points drawn, falling from 0.9 for a handful of points to 0.15 for 100,000 or more, so that overlapping points stay visible.
palette	Optional Character: Series colors, overriding the theme palette for this chart. NULL uses the theme's.
square	Logical: If TRUE, draw the plotting box square – equal height and width in pixels, excluding axis labels and margins.
equal_axes	Logical: If TRUE, give one data unit the same size in pixels on both axes. Set with square for a plot that is both, such as a true-versus-predicted plot whose identity line runs at 45 degrees; the two axes are then made to span the same interval.
pad	Numeric [0, Inf): Fraction of the data range to extend each axis by when xlim / ylim are not given. The default matches base R's xaxs = "r", which extends the range by 4% at each end.
xlim, ylim	Optional Numeric: Axis limits, length 2. NULL derives them from the data, padded by pad.
xlab, ylab	Optional Character: Axis labels. NULL derives them from the data.
margin_top, margin_right, margin_bottom, margin_left	Optional Integer [0, Inf): Plot margins in pixels.

Details

The data-binding properties (x, y, size, group) hold **column names**, not values. The values come from the data argument to draw(), or from the inherited dat_path.

Margins are declared as four scalars rather than one named vector: that is what states cleanly in a schema, while draw_scatter() keeps the convenient margins vector. Sides left NULL fall back to the chart's own layout.

Value

ScatterConfig object.

Author(s)

EDG

Examples

```
cfg <- setup_ScatterConfig(x = "wt", y = "mpg", fit = "glm")
cfg@type
```

`ScatterSeries` *Scatter Series*

Description

Configuration for a scatter (point) chart series.

Usage

```
ScatterSeries(
  name = NULL,
  data = NULL,
  x_axis_index = NULL,
  y_axis_index = NULL,
  silent = NULL,
  legend_hover_link = NULL,
  color = NULL,
  z_level = NULL,
  z = NULL,
  clip = NULL,
  symbol = NULL,
  symbol_size = NULL,
  symbol_rotate = NULL,
  symbol_offset = NULL,
  large = NULL,
  large_threshold = NULL,
  item_style = NULL,
  label = NULL
)
```

Arguments

`name`	Optional Character: Series name for legend display.
`data`	Optional Vector or list: Data values.
`x_axis_index`	Optional Numeric `[0, Inf)`: X-axis index.
`y_axis_index`	Optional Numeric `[0, Inf)`: Y-axis index.
`silent`	Optional Logical: Whether the series is silent.
`legend_hover_link`	Optional Logical: Whether to highlight related legends on hover.
`color`	Optional Character: Series color override.
`z_level`	Optional Numeric: Canvas layer index.

z	Optional Numeric: Front-back order within the same layer.
clip	Optional Logical: Whether to clip overflow.
symbol	Optional Character: Symbol type.
symbol_size	Optional Numeric or function: Symbol size.
symbol_rotate	Optional Numeric: Symbol rotation in degrees, counter-clockwise-positive. A symbol is unrotated at `0`.
symbol_offset	Optional Numeric or Character: Shift from the data point as `c(x, y)`, in screen pixels (numbers) or as a percentage of the symbol's own size (strings, e.g. `c("0%", "-50%")`). Applied *after* `symbol_rotate`, in the unrotated frame.
large	Optional Logical: Whether to enable large-data optimization.
large_threshold	Optional Numeric `[0, Inf)`: Threshold for large mode.
item_style	Optional ItemStyle: Data-point marker styling.
label	Optional LabelOption: Data-label configuration.

Details

Corresponds to `ScatterSeriesOption` in `src/chart/scatter/ScatterSeries.ts` (line 64). ECharts docs: https://echarts.apache.org/en/option.html#series-scatter

Value

`ScatterSeries` object.

Examples

```
ScatterSeries(name = "Points", data = list(c(1, 2), c(3, 4)), symbol_size = 8)
```

---

setup_A3Config *Set up an Annotated Protein Diagram Configuration*

Description

The seam between convenient input and a complete, validated object. **Every argument is optional**, which is what lets the published schema require nothing.

Usage

```
setup_A3Config(
  n_per_row = 21L,
  position_every = 10L,
  ptm_placement = "radial",
  zoom = TRUE,
  residue_spacing = 0.3,
  marker_size = 28,
```

```
  font_size = 18,
  line_width = 2,
  show_markers = TRUE,
  show_labels = TRUE,
  region_opacity = 0.3,
  residue_fill = NULL,
  residue_stroke = NULL,
  label_color = NULL,
  pos_label_color = NULL,
  variant_color = NULL,
  disease_variant_color = NULL,
  title = NULL,
  dat_path = NULL,
  origin = NULL,
  writer = NULL
)
```

Arguments

`n_per_row`	Integer `[1, Inf)`: Residues per row.
`position_every`	Integer `[1, Inf)`: Label every nth position.
`ptm_placement`	Character: How post-translational modifications are placed relative to the backbone.
`zoom`	Logical: Enable Shift+wheel zoom, and wheel and drag panning, along the sequence.
`residue_spacing`	Numeric `[0, Inf)`: Gap between residues.
`marker_size`	Numeric `[0, Inf)`: Preferred residue marker size; reduced with related marks when needed to fit the surface.
`font_size`	Numeric `[0, Inf)`: Residue label font size.
`line_width`	Numeric `[0, Inf)`: Backbone line width.
`show_markers, show_labels`	Logical: Whether to draw each.
`region_opacity`	Numeric `[0, 1]`: Region band opacity.
`residue_fill, residue_stroke`	Optional Character: Residue marker colors.
`label_color, pos_label_color`	Optional Character: Label colors.
`variant_color, disease_variant_color`	Optional Character: Variant marker colors. `NULL` selects the corresponding light/dark A3 theme color.
`title`	Optional Character: Chart title. Declared here because it is the one property every chart type has; `palette` is not (four take a `colormap` instead), and axis labels and margins are cartesian-only.
`dat_path`	Optional Character: Path to the data, read at draw time. The serializable alternative to passing `data` to `draw()`.

`origin`	Optional Named character: Where each value came from. Normally computed from which arguments were supplied; pass it only when restoring a config that already carries provenance.
`writer`	Optional Named character: Which interface wrote the config.

Value

A3Config object.

Author(s)

EDG

Examples

```
setup_A3Config(n_per_row = 30L)@n_per_row
```

`setup_BarConfig` *Set up a Bar Chart Configuration*

Description

The seam between convenient input and a complete, validated object. **Every argument is optional**, which is what lets the published schema require nothing.

Usage

```
setup_BarConfig(
  x = NULL,
  y = NULL,
  stack = FALSE,
  horizontal = FALSE,
  palette = NULL,
  xlab = NULL,
  ylab = NULL,
  title = NULL,
  margin_top = NULL,
  margin_right = NULL,
  margin_bottom = NULL,
  margin_left = NULL,
  dat_path = NULL,
  origin = NULL,
  writer = NULL,
  bar_width = NULL,
  order = "input",
  legend_position = "top",
  legend_placement = "outside",
```

```
  legend_title = NULL,
  zoom = FALSE
)
```

Arguments

x	Optional Character: Column holding the categories.
y	Optional Character: Columns holding the values, one series each.
stack	Logical: If TRUE, stack the series instead of grouping them.
horizontal	Logical: If TRUE, draw the bars horizontally.
palette	Optional Character: Series colors, overriding the theme palette for this chart. NULL uses the theme's.
xlab, ylab	Optional Character: Axis labels. NULL derives them from the data.
title	Optional Character: Chart title. Declared here because it is the one property every chart type has; palette is not (four take a colormap instead), and axis labels and margins are cartesian-only.
margin_top, margin_right, margin_bottom, margin_left	Optional Integer [0, Inf): Plot margins in pixels.
dat_path	Optional Character: Path to the data, read at draw time. The serializable alternative to passing data to draw().
origin	Optional Named character: Where each value came from. Normally computed from which arguments were supplied; pass it only when restoring a config that already carries provenance.
writer	Optional Named character: Which interface wrote the config.
bar_width	Optional Numeric (0, Inf): Bar thickness in pixels. NULL lets ECharts size bars from the category spacing.
order	Character {"input", "increasing", "decreasing"}: Order categories by their total across series, retaining ties and placing missing totals last.
legend_position	Character {"top", "bottom", "left", "right", "top-left", "top-right", "bottom-left", "bottom-right"}: Legend anchor. Top/bottom anchors use horizontal rows; left/right anchors use a vertical column. Corner anchors align within the top or bottom row.
legend_placement	Character {"outside", "inside"}: Relation to the plotting area. Outside placement reserves space for the complete legend; inside placement overlays the data. Neither setting adds a missing legend.
legend_title	Optional Character: Title drawn with the legend or colorbar: above left/right legends and vertical colorbars, before the first row of top/bottom legends and horizontal colorbars. NULL draws no title. A title does not add a missing legend.
zoom	Logical: Enable a slider, Shift+wheel zoom, and drag panning on the category axis.

Value

BarConfig object.

Author(s)

EDG

Examples

```
draw(setup_BarConfig(x = "cyl", y = "mpg"), data = aggregate(mpg ~ cyl, mtcars, mean))
```

---

setup_BoxplotConfig *Set up a Boxplot Configuration*

Description

The seam between convenient input and a complete, validated object. **Every argument is optional**, which is what lets the published schema require nothing.

Usage

```
setup_BoxplotConfig(
  x = NULL,
  group = NULL,
  labels = NULL,
  horizontal = FALSE,
  na_rm = TRUE,
  observation = NULL,
  quartiles = "linear",
  whisker = 1.5,
  boxpoints = "none",
  point_size = 5,
  point_alpha = 0.6,
  point_spread = 0.5,
  palette = NULL,
  fill_alpha = 0.25,
  xlab = NULL,
  ylab = NULL,
  title = NULL,
  margin_top = NULL,
  margin_right = NULL,
  margin_bottom = NULL,
  margin_left = NULL,
  dat_path = NULL,
  origin = NULL,
  writer = NULL,
  legend_position = "top",
  legend_placement = "outside",
  legend_title = NULL,
  geometry = "box",
  bandwidth = NULL,
```

```
  adjust = 1,
  density_points = 128L,
  paired = FALSE,
  pair_alpha = 0.35,
  pair_width = 1,
  comparisons = NULL,
  order = "input",
  transform = "none",
  zoom = FALSE
)
```

Arguments

x	Optional Character: Columns to summarize, one box each.
group	Optional Character: Column that splits each box by level.
labels	Optional Character: Box labels. NULL uses the bound column names.
horizontal	Logical: Draw the boxes horizontally.
na_rm	Logical: Remove missing values. FALSE rejects missing input.
observation	Optional Character: Column identifying observations for tooltips and pairing.
quartiles	Character {"linear", "hinges"}: Quartile convention.
whisker	Numeric [0, Inf): Finite IQR fence multiplier; zero uses full-range whiskers.
boxpoints	Character {"none", "all", "outliers"}: Values to overlay.
point_size	Numeric (0, Inf): Finite point diameter in pixels.
point_alpha	Numeric [0, 1]: Point opacity.
point_spread	Numeric [0, 1]: Fraction of box width occupied by offsets.
palette	Optional Character: Box colors, overriding the theme palette for this chart. NULL uses the package's box colors.
fill_alpha	Numeric [0, 1]: Box fill opacity.
xlab, ylab	Optional Character: Axis labels.
title	Optional Character: Chart title.
margin_top, margin_right, margin_bottom, margin_left	Optional Integer [0, Inf): Plot margins in pixels.
dat_path	Optional Character: Path to the data, read at draw time. The serializable alternative to passing data to draw().
origin	Optional Named character: Where each value came from. Normally computed from which arguments were supplied; pass it only when restoring a config that already carries provenance.
writer	Optional Named character: Which interface wrote the config.
legend_position	Character {"top", "bottom", "left", "right", "top-left", "top-right", "bottom-left", "bottom-right"}: Legend anchor. Top/bottom anchors use horizontal rows; left/right anchors use a vertical column. Corner anchors align within the top or bottom row.

`legend_placement`
: Character {"outside", "inside"}: Relation to the plotting area. Outside placement reserves space for the complete legend; inside placement overlays the data. Neither setting adds a missing legend.

`legend_title`
: Optional Character: Title drawn with the legend or colorbar: above left/right legends and vertical colorbars, before the first row of top/bottom legends and horizontal colorbars. `NULL` draws no title. A title does not add a missing legend.

`geometry`
: Character {"box", "violin", "both"}: Distribution geometry.

`bandwidth`
: Optional Numeric `(0, Inf)`: Finite Gaussian kernel bandwidth.

`adjust`
: Numeric `(0, Inf)`: Finite bandwidth multiplier.

`density_points`
: Integer `[16, 4096]`: Density evaluation grid size.

`paired`
: Logical: Connect explicit IDs across adjacent categories.

`pair_alpha`
: Numeric `[0, 1]`: Paired line opacity.

`pair_width`
: Numeric `(0, Inf)`: Finite paired line width in pixels.

`comparisons`
: Optional Character: Name of a comparison data frame in the bound data list. Its columns follow `draw_boxplot()`.

`order`
: Character `{"input", "mean", "median"}`: Keep input order or sort categories by decreasing mean/median after transformation; empty boxes last.

`transform`
: Character `{"none", "scale", "minmax"}`: Transform each input column before grouping. Scale subtracts the mean and divides by the sample standard deviation; minmax maps the available range to zero through one. Constant columns become zero and missing values remain missing.

`zoom`
: Logical: Enable a slider, Shift+wheel zoom, and drag panning on the category axis.

Value

BoxplotConfig object.

Author(s)

EDG

Examples

```
draw(setup_BoxplotConfig(x = c("mpg", "hp")), data = mtcars)
```

setup_CalibrationConfig *Set Up a Calibration Plot Configuration*

Description

Set Up a Calibration Plot Configuration

Usage

```
setup_CalibrationConfig(
  observed = "observed",
  probability = "probability",
  group = NULL,
  n_bins = 10L,
  bin_method = "quantile",
  na_rm = TRUE,
  mode = "lines+markers",
  show_brier = TRUE,
  rug = TRUE,
  rug_size = 8,
  rug_opacity = 0.3,
  point_size = 7,
  line_width = 2,
  digits = 3L,
  diagonal = TRUE,
  diagonal_color = "#888888",
  palette = NULL,
  legend = TRUE,
  legend_position = "top",
  legend_placement = "outside",
  legend_title = NULL,
  square = TRUE,
  xlab = "Mean predicted probability",
  ylab = "Observed proportion",
  title = NULL,
  dat_path = NULL,
  origin = NULL,
  writer = NULL
)
```

Arguments

observed	Character: Column containing binary outcomes, zero or one.
probability	Character: Column containing predicted probabilities.
group	Optional Character: Column identifying samples or models.

n_bins	Integer [1, Inf): Requested number of bins per group.
bin_method	Character {"quantile", "equidistant"}: Binning rule.
na_rm	Logical: Remove incomplete observation/probability pairs.
mode	Character {"lines", "markers", "lines+markers"}: Curve display.
show_brier	Logical: Include each sample's Brier score in its legend label.
rug	Logical: Show the distribution of individual probabilities.
rug_size	Numeric (0, Inf): Rug tick height in pixels.
rug_opacity	Numeric [0, 1]: Rug tick opacity.
point_size	Numeric (0, Inf): Calibration marker diameter in pixels.
line_width	Numeric: Curve stroke width in pixels.
digits	Integer: Decimal places for AUC labels and tooltip values.
diagonal	Logical: Show an independent chance diagonal.
diagonal_color	Character: Chance-line color.
palette	Optional Character: Group colors; unset uses the chart theme.
legend	Logical: Show group labels and AUC summaries.
legend_position	Character {"top", "bottom", "left", "right", "top-left", "top-right", "bottom-left", "bottom-right"}: Legend anchor. Top/bottom anchors use horizontal rows; left/right anchors use a vertical column. Corner anchors align within the top or bottom row.
legend_placement	Character {"outside", "inside"}: Relation to the plotting area. Outside placement reserves space for the complete legend; inside placement overlays the data. Neither setting adds a missing legend.
legend_title	Optional Character: Title drawn with the legend or colorbar: above left/right legends and vertical colorbars, before the first row of top/bottom legends and horizontal colorbars. NULL draws no title. A title does not add a missing legend.
square	Logical: Keep the plotting grid square.
xlab	Character: Horizontal axis label.
ylab	Character: Vertical axis label.
title	Optional Character: Chart title.
dat_path	Optional Character: Path to the data, read at draw time. The serializable alternative to passing data to draw().
origin	Optional Named character: Per-property provenance.
writer	Optional Named character: Interface name and version.

Value

A CalibrationConfig object.

Examples

```
setup_CalibrationConfig(group = "model", n_bins = 5L)
```

setup_ChoroplethConfig *Set up a Choropleth Map Configuration*

Description

The seam between convenient input and a complete, validated object. **Every argument is optional**, which is what lets the published schema require nothing.

Usage

```
setup_ChoroplethConfig(
  location = NULL,
  value = NULL,
  tooltip = NULL,
  resolution = "country",
  classification = "quantile",
  num_classes = 5,
  colormap = "blues",
  opacity = 1,
  show_boundaries = TRUE,
  outline_width = 0.2,
  show_legend = TRUE,
  legend_position = "bottom-right",
  tooltip_position = "top-right",
  value_label = NULL,
  title = NULL,
  dat_path = NULL,
  origin = NULL,
  writer = NULL
)
```

Arguments

`location`	Optional Character: Column holding the region identifiers.
`value`	Optional Character: Column holding the value to shade by.
`tooltip`	Optional Character: Column holding per-region tooltip text.
`resolution`	Character {"country", "state", "county"}: Which boundary set the locations refer to.
`classification`	Character {"quantile", "equal", "jenks"}: How values are binned into classes.
`num_classes`	Numeric `[2, 12]`: Number of color classes.
`colormap`	Character: Continuous color scheme for the classes.
`opacity`	Numeric `[0, 1]`: Fill opacity.
`show_boundaries, show_legend`	Logical: Whether to draw each.

`outline_width` Numeric `[0, Inf)`: Boundary line width.

`legend_position, tooltip_position`
Character: Corner each overlay is anchored to.

`value_label` Optional Character: Label for the value in the legend and tooltip. `NULL` derives it from the bound column.

`title` Optional Character: Chart title. Declared here because it is the one property every chart type has; `palette` is not (four take a `colormap` instead), and axis labels and margins are cartesian-only.

`dat_path` Optional Character: Path to the data, read at draw time. The serializable alternative to passing `data` to `draw()`.

`origin` Optional Named character: Where each value came from. Normally computed from which arguments were supplied; pass it only when restoring a config that already carries provenance.

`writer` Optional Named character: Which interface wrote the config.

Value

ChoroplethConfig object.

Author(s)

EDG

Examples

```
states <- data.frame(st = c("CA", "TX", "NY"), v = c(1, 2, 3))
draw(
  setup_ChoroplethConfig(location = "st", value = "v", resolution = "state"),
  data = states
)
```

---

setup_ConfusionConfig *Set Up a Confusion Plot Configuration*

Description

Set Up a Confusion Plot Configuration

Usage

```
setup_ConfusionConfig(
  reference = "reference",
  predicted = "predicted",
  count = "n",
  panel = NULL,
  classes = NULL,
```

```
  show_metrics = TRUE,
  digits = 2L,
  ncol = 2L,
  correct_color = "#0F6A66",
  incorrect_color = "#BE2E5F",
  low_color = NULL,
  summary_color = NULL,
  font_size = 12,
  xlab = "Predicted",
  ylab = "Reference",
  title = NULL,
  dat_path = NULL,
  origin = NULL,
  writer = NULL
)
```

Arguments

reference	Character: Column containing reference class labels.
predicted	Character: Column containing predicted class labels.
count	Character: Column containing nonnegative integer frequencies.
panel	Optional Character: Optional panel-label column; unset draws one matrix.
classes	Optional Character vector: Ordered class labels shared by every panel; unset preserves first appearance.
show_metrics	Logical: Show per-class rates, accuracy, and balanced accuracy.
digits	Integer: Decimal places for rates in cells and hover text.
ncol	Integer: Maximum number of panels per row.
correct_color	Character: Six-digit hex color at unit fraction for correct predictions.
incorrect_color	Character: Six-digit hex color at unit fraction for incorrect predictions.
low_color	Optional Character: Six-digit hex color at zero row fraction; unset uses the active theme background.
summary_color	Optional Character: Six-digit hex background for metric cells; unset uses a faint neutral tint of the active theme background.
font_size	Numeric: Preferred cell-label font size in pixels. Constrained surfaces reduce text to fit, down to 8 pixels. Long class labels wrap and column labels rotate when needed. Increase figure dimensions if labels and metrics cannot fit without overlap.
xlab	Character: Predicted-class axis label.
ylab	Character: Reference-class axis label.
title	Optional Character: Chart title. Declared here because it is the one property every chart type has; palette is not (four take a colormap instead), and axis labels and margins are cartesian-only.

dat_path	Optional Character: Path to the data, read at draw time. The serializable alternative to passing data to draw().
origin	Optional Named character: Per-property provenance.
writer	Optional Named character: Interface name and version.

Value

A ConfusionConfig object.

Examples

```
setup_ConfusionConfig(panel = "split", classes = c("yes", "no"))
```

---

setup_DensityConfig *Set up a Density Chart Configuration*

Description

The seam between convenient input and a complete, validated object. **Every argument is optional**, which is what lets the published schema require nothing.

Usage

```
setup_DensityConfig(
  x = NULL,
  group = NULL,
  n = 512L,
  bw = "nrd0",
  na_rm = TRUE,
  palette = NULL,
  xlab = NULL,
  ylab = NULL,
  title = NULL,
  margin_top = NULL,
  margin_right = NULL,
  margin_bottom = NULL,
  margin_left = NULL,
  dat_path = NULL,
  origin = NULL,
  writer = NULL,
  legend_position = "top",
  legend_placement = "outside",
  legend_title = NULL,
  bandwidth = NULL,
  kernel = "gaussian",
  adjust = 1,
  fill_alpha = 0.25,
```

```
  mode = "overlap",
  order = "input",
  zoom = FALSE
)
```

Arguments

x	Optional Character: Columns to estimate, one distribution per column.
group	Optional Character: Column to split the estimate by.
n	Integer [2, Inf): Points at which the density is estimated.
bw	Character or Numeric: Bandwidth selector ("nrd0", "nrd", "ucv", "bcv", "SJ", "SJ-ste", "SJ-dpi"). Numeric input is normalized to bandwidth by the setup function.
na_rm	Logical: If TRUE, drop NA values before estimating.
palette	Optional Character: Series colors, overriding the theme palette for this chart. NULL uses the theme's.
xlab, ylab	Optional Character: Axis labels. NULL derives them from the data.
title	Optional Character: Chart title. Declared here because it is the one property every chart type has; palette is not (four take a colormap instead), and axis labels and margins are cartesian-only.
margin_top, margin_right, margin_bottom, margin_left	Optional Integer [0, Inf): Plot margins in pixels.
dat_path	Optional Character: Path to the data, read at draw time. The serializable alternative to passing data to draw().
origin	Optional Named character: Where each value came from. Normally computed from which arguments were supplied; pass it only when restoring a config that already carries provenance.
writer	Optional Named character: Which interface wrote the config.
legend_position	Character {"top", "bottom", "left", "right", "top-left", "top-right", "bottom-left", "bottom-right"}: Legend anchor. Top/bottom anchors use horizontal rows; left/right anchors use a vertical column. Corner anchors align within the top or bottom row.
legend_placement	Character {"outside", "inside"}: Relation to the plotting area. Outside placement reserves space for the complete legend; inside placement overlays the data. Neither setting adds a missing legend.
legend_title	Optional Character: Title drawn with the legend or colorbar: above left/right legends and vertical colorbars, before the first row of top/bottom legends and horizontal colorbars. NULL draws no title. A title does not add a missing legend.
bandwidth	Optional Numeric (0, Inf): Explicit finite smoothing bandwidth in measurement units, overriding the selector.
kernel	Character: One of "gaussian", "epanechnikov", "rectangular", "triangular", "biweight", "cosine", or "optcosine".

`adjust`	Numeric `(0, Inf)`: Finite bandwidth multiplier.
`fill_alpha`	Numeric `[0, 1]`: Distribution fill opacity.
`mode`	Character `{"overlap", "ridge"}`: Overlay samples or align rows on common x and y scales. Ridge rows use sample names in place of a legend. Increase figure height for many rows.
`order`	Character `{"input", "mean", "median"}`: Sample order. Summary orders are decreasing, retain ties in input order, and place empty samples last.
`zoom`	Logical: Enable a slider, Shift+wheel zoom, and drag panning on the x-axis. Ridge rows zoom together, without a slider.

Value

DensityConfig object.

Author(s)

EDG

Examples

```
draw(setup_DensityConfig(x = "mpg"), data = mtcars)
```

`setup_GanttConfig` *Set up a Gantt Chart Configuration*

Description

The seam between convenient input and a complete, validated object. **Every argument is optional**, which is what lets the published schema require nothing.

Usage

```
setup_GanttConfig(
  label = "label",
  start = "start",
  end = "end",
  group = NULL,
  tooltip = NULL,
  border = NULL,
  axis_type = "value",
  zoom = TRUE,
  guides = TRUE,
  bar_height = 0.6,
  bar_radius = 0,
  border_color = "#E53935",
  border_width = 1.5,
```

```
  palette = NULL,
  xlab = NULL,
  title = NULL,
  dat_path = NULL,
  origin = NULL,
  writer = NULL,
  legend_position = "top",
  legend_placement = "outside",
  legend_title = NULL
)
```

Arguments

label	Character: Column naming each task.
start	Character: Column holding each task's start.
end	Character: Column holding each task's end.
group	Optional Character: Column to color bars by.
tooltip	Optional Character: Column holding per-bar tooltip text.
border	Optional Character: Logical column; TRUE outlines the bar.
axis_type	Character {"value", "time"}: How the time axis is scaled.
zoom	Logical: Enable Shift+wheel zoom, drag panning, and box-zoom controls on both axes.
guides	Logical: Show guide lines.
bar_height	Numeric [0, 1]: Bar height as a fraction of the row.
bar_radius	Numeric [0, Inf): Bar corner radius in pixels.
border_color	Character: Outline color for bordered bars.
border_width	Numeric [0, Inf): Outline width in pixels.
palette	Optional Character: Bar colors, overriding the theme palette for this chart. NULL uses the theme's.
xlab	Optional Character: Time axis label.
title	Optional Character: Chart title. Declared here because it is the one property every chart type has; palette is not (four take a colormap instead), and axis labels and margins are cartesian-only.
dat_path	Optional Character: Path to the data, read at draw time. The serializable alternative to passing data to draw().
origin	Optional Named character: Where each value came from. Normally computed from which arguments were supplied; pass it only when restoring a config that already carries provenance.
writer	Optional Named character: Which interface wrote the config.
legend_position	Character {"top", "bottom", "left", "right", "top-left", "top-right", "bottom-left", "bottom-right"}: Legend anchor. Top/bottom anchors use horizontal rows; left/right anchors use a vertical column. Corner anchors align within the top or bottom row.

`legend_placement`
: Character {"outside", "inside"}: Relation to the plotting area. Outside placement reserves space for the complete legend; inside placement overlays the data. Neither setting adds a missing legend.

`legend_title`
: Optional Character: Title drawn with the legend or colorbar: above left/right legends and vertical colorbars, before the first row of top/bottom legends and horizontal colorbars. `NULL` draws no title. A title does not add a missing legend.

Value

GanttConfig object.

Author(s)

EDG

Examples

```
tasks <- data.frame(
  label = c("s1", "s2"),
  start = c(0, 5),
  end = c(5, 9),
  kind = c("a", "b")
)
draw(setup_GanttConfig(group = "kind"), data = tasks)
```

---

setup_HeatmapConfig *Set up a Heatmap Configuration*

Description

The seam between convenient input and a complete, validated object. **Every argument is optional**, which is what lets the published schema require nothing.

Usage

```
setup_HeatmapConfig(
  row_names = NULL,
  col_names = NULL,
  triangle = NULL,
  cluster_rows = FALSE,
  cluster_cols = FALSE,
  dist_method = "euclidean",
  hclust_method = "complete",
  square_cells = NULL,
  show_row_dendro = TRUE,
  show_col_dendro = TRUE,
  dendro_row_width = 60,
```

```
  dendro_col_height = 60,
  dendro_color = NULL,
  dendro_uniform = FALSE,
  dendro_row_side = "right",
  dendro_col_side = "top",
  colormap = NULL,
  zlim = NULL,
  show_notes = FALSE,
  row_cut = NULL,
  col_cut = NULL,
  show_values = FALSE,
  value_digits = 2L,
  show_colorbar = TRUE,
  colorbar_orient = "vertical",
  title = NULL,
  margin_top = NULL,
  margin_right = NULL,
  margin_bottom = NULL,
  margin_left = NULL,
  dat_path = NULL,
  origin = NULL,
  writer = NULL,
  legend_position = if (colorbar_orient == "horizontal") "bottom" else "right",
  legend_placement = "outside",
  legend_title = NULL
)
```

Arguments

`row_names`, `col_names`	Optional Character: Labels overriding the matrix dimnames.
`triangle`	Optional Character {"lower", "upper"}: Show only one triangle, for a symmetric matrix.
`cluster_rows`, `cluster_cols`	Logical: Reorder by hierarchical clustering.
`dist_method`	Character: Distance measure for clustering.
`hclust_method`	Character: Linkage method for clustering.
`square_cells`	Optional Logical: Force square cells, sizing the container to match.
`show_row_dendro`, `show_col_dendro`	Logical: Draw each dendrogram.
`dendro_row_width`, `dendro_col_height`	Numeric `[0, Inf)`: Dendrogram panel size in pixels.
`dendro_color`	Optional Character: Dendrogram line color.
`dendro_uniform`	Logical: Draw dendrograms with uniform branch heights.
`dendro_row_side`	Character {"left", "right"}: Row dendrogram side.

`dendro_col_side`
: Character {"top", "bottom"}: Column dendrogram side.

`colormap`
: Optional Character: Colors defining the continuous scale. `NULL` derives a theme-aware diverging or sequential scale.

`zlim`
: Optional Numeric: Color-scale limits, length 2.

`show_notes`
: Logical: Print supplied cell notes instead of numeric values.

`row_cut`, `col_cut`
: Optional Integer: Number of clusters for branch coloring, requiring a computed or supplied tree.

`show_values`
: Logical: Print each cell's value.

`value_digits`
: Integer `[0, Inf)`: Digits for printed values.

`show_colorbar`
: Logical: Draw the color bar.

`colorbar_orient`
: Character {"vertical", "horizontal"}: Color bar orientation.

`title`
: Optional Character: Chart title. Declared here because it is the one property every chart type has; `palette` is not (four take a `colormap` instead), and axis labels and margins are cartesian-only.

`margin_top`, `margin_right`, `margin_bottom`, `margin_left`
: Optional Integer `[0, Inf)`: Override the auto-computed margins, per side.

`dat_path`
: Optional Character: Path to the data, read at draw time. The serializable alternative to passing `data` to `draw()`.

`origin`
: Optional Named character: Where each value came from. Normally computed from which arguments were supplied; pass it only when restoring a config that already carries provenance.

`writer`
: Optional Named character: Which interface wrote the config.

`legend_position`
: Character {"top", "bottom", "left", "right", "top-left", "top-right", "bottom-left", "bottom-right"}: Legend anchor. Top/bottom anchors use horizontal rows; left/right anchors use a vertical column. Corner anchors align within the top or bottom row.

`legend_placement`
: Character {"outside", "inside"}: Relation to the plotting area. Outside placement reserves space for the complete legend; inside placement overlays the data. Neither setting adds a missing legend.

`legend_title`
: Optional Character: Title drawn with the legend or colorbar: above left/right legends and vertical colorbars, before the first row of top/bottom legends and horizontal colorbars. `NULL` draws no title. A title does not add a missing legend.

Value

HeatmapConfig object.

Author(s)

EDG

Examples

```
draw(setup_HeatmapConfig(), data = matrix(1:20, nrow = 4))
```

`setup_HistogramConfig` *Set up a Histogram Configuration*

Description

The seam between convenient input and a complete, validated object. **Every argument is optional**, which is what lets the published schema require nothing.

Usage

```
setup_HistogramConfig(
  x = NULL,
  group = NULL,
  breaks = "Sturges",
  palette = NULL,
  xlab = NULL,
  ylab = NULL,
  title = NULL,
  margin_top = NULL,
  margin_right = NULL,
  margin_bottom = NULL,
  margin_left = NULL,
  dat_path = NULL,
  origin = NULL,
  writer = NULL,
  legend_position = "top",
  legend_placement = "outside",
  legend_title = NULL,
  bins = NULL,
  bin_edges = NULL,
  normalization = "count",
  density = FALSE,
  n = 512L,
  bw = "nrd0",
  bandwidth = NULL,
  kernel = "gaussian",
  adjust = 1,
  fill_alpha = NULL,
  border_alpha = 1,
  na_rm = TRUE,
  mode = "overlap",
  order = "input",
  bin_stat = "count",
```

```
  bar_mode = "overlay",
  zoom = FALSE
)
```

Arguments

x	Optional Character: Columns to bin, one distribution per column.
group	Optional Character: Column to split the bins by.
breaks	Numeric, Character, or Numeric vector: Binning method. A single number (number of bins), a character string naming an algorithm (e.g. "Sturges", "Scott", "FD"), or a numeric vector of break points. Numeric forms are normalized to bins or bin_edges by the setup function. Bin counts are suggestions to hist().
palette	Optional Character: Series colors, overriding the theme palette for this chart. NULL uses the theme's.
xlab, ylab	Optional Character: Axis labels. NULL derives them from the data.
title	Optional Character: Chart title. Declared here because it is the one property every chart type has; palette is not (four take a colormap instead), and axis labels and margins are cartesian-only.
margin_top, margin_right, margin_bottom, margin_left	Optional Integer [0, Inf): Plot margins in pixels.
dat_path	Optional Character: Path to the data, read at draw time. The serializable alternative to passing data to draw().
origin	Optional Named character: Where each value came from. Normally computed from which arguments were supplied; pass it only when restoring a config that already carries provenance.
writer	Optional Named character: Which interface wrote the config.
legend_position	Character {"top", "bottom", "left", "right", "top-left", "top-right", "bottom-left", "bottom-right"}: Legend anchor. Top/bottom anchors use horizontal rows; left/right anchors use a vertical column. Corner anchors align within the top or bottom row.
legend_placement	Character {"outside", "inside"}: Relation to the plotting area. Outside placement reserves space for the complete legend; inside placement overlays the data. Neither setting adds a missing legend.
legend_title	Optional Character: Title drawn with the legend or colorbar: above left/right legends and vertical colorbars, before the first row of top/bottom legends and horizontal colorbars. NULL draws no title. A title does not add a missing legend.
bins	Optional Integer [1, 1000000]: Suggested bin count.
bin_edges	Optional Numeric vector: Finite, strictly increasing edges spanning every retained observation. Overrides breaks; excludes bins.
normalization	Character: "count", "probability", "percent", "density", or "count_density".
density	Logical: Overlay a kernel density in histogram units.

n	Integer [2, Inf): Number of density evaluation points.
bw	Character or Numeric: Bandwidth selector ("nrd0", "nrd", "ucv", "bcv", "SJ", "SJ-ste", "SJ-dpi"). Numeric input is normalized to bandwidth by the setup function.
bandwidth	Optional Numeric (0, Inf): Explicit finite smoothing bandwidth in measurement units, overriding the selector.
kernel	Character: One of "gaussian", "epanechnikov", "rectangular", "triangular", "biweight", "cosine", or "optcosine".
adjust	Numeric (0, Inf): Finite bandwidth multiplier.
fill_alpha	Optional Numeric [0, 1]: Bar fill opacity. NULL uses 0.5 when overlaid groups overlap (bar_mode = "overlay" with more than one nonempty group outside ridge mode) and 0.75 otherwise.
border_alpha	Numeric [0, 1]: Opacity of the bar borders, drawn in the series color. 0 draws no border.
na_rm	Logical: Whether to remove NA values before computing densities.
mode	Character {"overlap", "ridge"}: Overlay samples or align rows on common x and y scales. Ridge rows use sample names in place of a legend. Increase figure height for many rows.
order	Character {"input", "mean", "median"}: Sample order. Summary orders are decreasing, retain ties in input order, and place empty samples last.
bin_stat	Character {"count", "sum", "mean", "min", "max"}: Statistic of x observations in each bin. Empty bins have height zero. Non-count statistics require normalization = "count" and density = FALSE.
bar_mode	Character {"overlay", "group", "stack"}: Histogram group layout. Stacks accumulate positive and negative values separately. Group/stack layouts require mode = "overlap" and density = FALSE.
zoom	Logical: Enable a slider, Shift+wheel zoom, and drag panning on the x-axis. Ridge rows zoom together, without a slider.

Value

HistogramConfig object.

Author(s)

EDG

Examples

```
draw(setup_HistogramConfig(x = "mpg"), data = mtcars)
```

`setup_LineConfig` *Set up a Line Chart Configuration*

Description

The seam between convenient input and a complete, validated object. **Every argument is optional**, which is what lets the published schema require nothing.

Usage

```
setup_LineConfig(
  x = NULL,
  y = NULL,
  group = NULL,
  blocks = NULL,
  smooth = FALSE,
  area = FALSE,
  points = TRUE,
  zoom = FALSE,
  palette = NULL,
  block_color = NULL,
  block_opacity = 0.2,
  pad = DEFAULT_PAD,
  square = FALSE,
  equal_axes = FALSE,
  xlim = NULL,
  ylim = NULL,
  xlab = NULL,
  ylab = NULL,
  title = NULL,
  margin_top = NULL,
  margin_right = NULL,
  margin_bottom = NULL,
  margin_left = NULL,
  dat_path = NULL,
  origin = NULL,
  writer = NULL,
  legend_position = "top",
  legend_placement = "outside",
  legend_title = NULL
)
```

Arguments

`x`	Optional Character: Column along the horizontal axis. A numeric column gets a value axis, a `Date` or `POSIXct` column a time axis with adaptively chosen labels, and any other column a category axis.

y	Optional Character: Columns to plot, one line each.
group	Optional Character: Column to group and color lines by. Splits a single y column into one line per level; cannot be combined with several y columns or with blocks.
blocks	Optional Character: Column whose contiguous runs shade vertical background bands.
smooth	Logical: Draw smoothed lines.
area	Logical: Fill the area under each line.
points	Logical: Show a marker at each data value.
zoom	Logical: Enable an x-axis slider, Shift+wheel zoom, and drag panning.
palette	Optional Character: Series colors, overriding the theme palette for this chart. NULL uses the theme's.
block_color	Optional Character: Band colors.
block_opacity	Numeric [0, 1]: Band opacity.
pad	Numeric [0, Inf): Fraction of the data range to extend each axis by when the limits are not given.
square	Logical: If TRUE, draw the plotting box square – equal height and width in pixels, excluding axis labels and margins.
equal_axes	Logical: If TRUE, give one data unit the same size in pixels on both axes. Set with square for a plot that is both, such as an ROC curve; the two axes are then made to span the same interval. Requires a numeric x.
xlim, ylim	Optional Numeric: Axis limits, length 2. NULL derives them from the data, padded by pad. On a time axis xlim is in epoch milliseconds, the unit the chart positions by.
xlab, ylab	Optional Character: Axis labels. NULL derives them from the data.
title	Optional Character: Chart title. Declared here because it is the one property every chart type has; palette is not (four take a colormap instead), and axis labels and margins are cartesian-only.
margin_top, margin_right, margin_bottom, margin_left	Optional Integer [0, Inf): Plot margins in pixels.
dat_path	Optional Character: Path to the data, read at draw time. The serializable alternative to passing data to draw().
origin	Optional Named character: Where each value came from. Normally computed from which arguments were supplied; pass it only when restoring a config that already carries provenance.
writer	Optional Named character: Which interface wrote the config.
legend_position	Character {"top", "bottom", "left", "right", "top-left", "top-right", "bottom-left", "bottom-right"}: Legend anchor. Top/bottom anchors use horizontal rows; left/right anchors use a vertical column. Corner anchors align within the top or bottom row.

`legend_placement`
: Character {"outside", "inside"}: Relation to the plotting area. Outside placement reserves space for the complete legend; inside placement overlays the data. Neither setting adds a missing legend.

`legend_title`
: Optional Character: Title drawn with the legend or colorbar: above left/right legends and vertical colorbars, before the first row of top/bottom legends and horizontal colorbars. `NULL` draws no title. A title does not add a missing legend.

Value

LineConfig object.

Author(s)

EDG

Examples

```
draw(setup_LineConfig(x = "wt", y = "mpg"), data = mtcars[order(mtcars$wt), ])
```

---

setup_MermaidConfig *Set up a Mermaid Diagram Configuration*

Description

The seam between convenient input and a complete, validated object. **Every argument is optional**, which is what lets the published schema require nothing.

Usage

```
setup_MermaidConfig(
  palette = NULL,
  node_color = NULL,
  fill_alpha = 0.1,
  font_size = 14,
  padding = NULL,
  auto_direction = FALSE,
  fit = FALSE,
  title = NULL,
  dat_path = NULL,
  origin = NULL,
  writer = NULL
)
```

Arguments

palette	Optional Character: Colors for categorical entities, overriding the theme palette. NULL uses the theme's.
node_color	Optional Named character: Colors for selected nodes, named by node id. Each named node gets a border in its color and a fill of the same color at fill_alpha opacity.
fill_alpha	Numeric [0, 1]: Fill opacity of the nodes in node_color.
font_size	Numeric (0, Inf): Label font size in pixels.
padding	Optional Numeric [0, Inf): Space around the diagram in pixels. NULL uses Pele's default.
auto_direction	Logical: Allow left-to-right flowcharts to be drawn top-to-bottom in widgets narrower than 640 pixels.
fit	Logical: Shrink the diagram to the widget width instead of drawing it at its natural size.
title	Optional Character: Chart title. Declared here because it is the one property every chart type has; palette is not (four take a colormap instead), and axis labels and margins are cartesian-only.
dat_path	Optional Character: Path to the data, read at draw time. The serializable alternative to passing data to draw().
origin	Optional Named character: Where each value came from. Normally computed from which arguments were supplied; pass it only when restoring a config that already carries provenance.
writer	Optional Named character: Which interface wrote the config.

Value

MermaidConfig object.

Author(s)

EDG

Examples

```
config <- setup_MermaidConfig(title = "Study design")
draw(
  config,
  data = c(
    "flowchart TD",
    "  screened[Screened] --> eligible[Eligible]",
    "  eligible --> treated[Treatment]",
    "  eligible --> control[Control]"
  )
)
```

`setup_NetworkConfig` *Set up a Network Graph Configuration*

Description

The seam between convenient input and a complete, validated object. **Every argument is optional**, which is what lets the published schema require nothing.

Usage

```
setup_NetworkConfig(
  directed = FALSE,
  threshold = NULL,
  self_loops = FALSE,
  layout = "force",
  cluster_resolution = 1,
  scale_by_degree = TRUE,
  color_by_group = FALSE,
  blend_edges = FALSE,
  show_labels = TRUE,
  node_size = 10,
  edge_scale = 3,
  node_opacity = 0.95,
  edge_opacity = 0.4,
  palette = NULL,
  node_color = rtemis_colors[["teal"]],
  positive_color = NULL,
  negative_color = NULL,
  title = NULL,
  dat_path = NULL,
  origin = NULL,
  writer = NULL
)
```

Arguments

`directed`	Logical: Treat the matrix as directed.
`threshold`	Optional Numeric: Drop edges weaker than this.
`self_loops`	Logical: Keep self-edges.
`layout`	Character: Layout algorithm.
`cluster_resolution`	Numeric `[0, Inf)`: Community-detection resolution. Higher values find more, smaller communities.
`scale_by_degree`	Logical: Size nodes by their degree.
`color_by_group`	Logical: Color nodes by detected community.

`blend_edges` Logical: Blend overlapping edges.

`show_labels` Logical: Draw node labels.

`node_size` Numeric `[0, Inf)`: Base node size.

`edge_scale` Numeric `[0, Inf)`: Edge width multiplier.

`node_opacity`, `edge_opacity`
Numeric `[0, 1]`: Opacity of each.

`palette` Optional Character: Community colors, overriding the theme palette. `NULL` uses the theme's.

`node_color` Character: Node color when not colored by group.

`positive_color`, `negative_color`
Optional Character: Edge colors by sign. `NULL` uses the theme's sign colors.

`title` Optional Character: Chart title. Declared here because it is the one property every chart type has; `palette` is not (four take a `colormap` instead), and axis labels and margins are cartesian-only.

`dat_path` Optional Character: Path to the data, read at draw time. The serializable alternative to passing `data` to `draw()`.

`origin` Optional Named character: Where each value came from. Normally computed from which arguments were supplied; pass it only when restoring a config that already carries provenance.

`writer` Optional Named character: Which interface wrote the config.

Value

NetworkConfig object.

Author(s)

EDG

Examples

```
m <- matrix(
  c(0, 1, 1, 1, 0, 1, 1, 1, 0),
  nrow = 3,
  dimnames = list(letters[1:3], letters[1:3])
)
draw(setup_NetworkConfig(), data = m)
```

setup_PanelLayout *Set Up a Panel Layout*

Description

Set Up a Panel Layout

Usage

```
setup_PanelLayout(ncol = 1L, gap = 12, padding = 0)
```

Arguments

ncol	Integer [1, Inf): Number of columns.
gap	Numeric [0, Inf): Finite gap between cells in pixels.
padding	Numeric [0, Inf): Finite outer padding in pixels.

Value

PanelLayout: Validated settings.

Examples

```
setup_PanelLayout(ncol = 2, gap = 16)
```

setup_PieConfig *Set up a Pie Chart Configuration*

Description

The seam between convenient input and a complete, validated object. **Every argument is optional**, which is what lets the published schema require nothing.

Usage

```
setup_PieConfig(
  values = NULL,
  labels = NULL,
  radius = "75%",
  rose_type = NULL,
  label_format = "name",
  percent_digits = 1L,
  palette = NULL,
  title = NULL,
  dat_path = NULL,
```

```
  origin = NULL,
  writer = NULL,
  legend_position = "top",
  legend_placement = "outside",
  legend_title = NULL
)
```

Arguments

values	Optional Character: Column holding the slice values.
labels	Optional Character: Column holding the slice labels.
radius	Character: Outer radius, as a CSS length or percentage.
rose_type	Optional Character {"radius", "area"}: Draw as a Nightingale rose chart, sizing slices by radius or by area. NULL draws a plain pie.
label_format	Character {"name", "value", "percent", "name_value", "name_percent"}: Slice label content.
percent_digits	Integer [0, 6]: Percentage precision.
palette	Optional Character: Slice colors, overriding the theme palette for this chart. NULL uses the theme's.
title	Optional Character: Chart title. Declared here because it is the one property every chart type has; palette is not (four take a colormap instead), and axis labels and margins are cartesian-only.
dat_path	Optional Character: Path to the data, read at draw time. The serializable alternative to passing data to draw().
origin	Optional Named character: Where each value came from. Normally computed from which arguments were supplied; pass it only when restoring a config that already carries provenance.
writer	Optional Named character: Which interface wrote the config.
legend_position	Character {"top", "bottom", "left", "right", "top-left", "top-right", "bottom-left", "bottom-right"}: Legend anchor. Top/bottom anchors use horizontal rows; left/right anchors use a vertical column. Corner anchors align within the top or bottom row.
legend_placement	Character {"outside", "inside"}: Relation to the plotting area. Outside placement reserves space for the complete legend; inside placement overlays the data. Neither setting adds a missing legend.
legend_title	Optional Character: Title drawn with the legend or colorbar: above left/right legends and vertical colorbars, before the first row of top/bottom legends and horizontal colorbars. NULL draws no title. A title does not add a missing legend.

Value

PieConfig object.

Author(s)

EDG

Examples

```
counts <- data.frame(kind = c("a", "b", "c"), n = c(3, 5, 2))
draw(setup_PieConfig(values = "n", labels = "kind"), data = counts)
```

---

setup_ROCConfig *Set Up a ROC Plot Configuration*

Description

Set Up a ROC Plot Configuration

Usage

```
setup_ROCConfig(
  fpr = "fpr",
  tpr = "tpr",
  auc = NULL,
  class_label = NULL,
  split = NULL,
  fold = NULL,
  omitted = NULL,
  variant = "aggregate",
  digits = 3L,
  diagonal = TRUE,
  diagonal_color = "#888888",
  palette = NULL,
  legend = TRUE,
  legend_position = "top",
  legend_placement = "outside",
  legend_title = NULL,
  square = TRUE,
  line_width = 2,
  fold_opacity = 0.45,
  xlab = "False positive rate",
  ylab = "True positive rate",
  title = NULL,
  dat_path = NULL,
  origin = NULL,
  writer = NULL
)
```

Arguments

fpr	Character: Column containing false positive rates.
tpr	Character: Column containing true positive rates.
auc	Optional Character: Optional supplied AUC column; unset integrates the supplied vertices.
class_label	Optional Character: Optional positive-class label column.
split	Optional Character: Optional sample or model label column.
fold	Optional Character: Optional resample column; aggregate identifies pooled curves.
omitted	Optional Character: Optional omitted-observation count column, constant within each curve.
variant	Character: Draw pooled curves or separate resample curves.
digits	Integer: Decimal places for AUC labels and tooltip values.
diagonal	Logical: Show an independent chance diagonal.
diagonal_color	Character: Chance-line color.
palette	Optional Character: Group colors; unset uses the chart theme.
legend	Logical: Show group labels and AUC summaries.
legend_position	Character {"top", "bottom", "left", "right", "top-left", "top-right", "bottom-left", "bottom-right"}: Legend anchor. Top/bottom anchors use horizontal rows; left/right anchors use a vertical column. Corner anchors align within the top or bottom row.
legend_placement	Character {"outside", "inside"}: Relation to the plotting area. Outside placement reserves space for the complete legend; inside placement overlays the data. Neither setting adds a missing legend.
legend_title	Optional Character: Title drawn with the legend or colorbar: above left/right legends and vertical colorbars, before the first row of top/bottom legends and horizontal colorbars. NULL draws no title. A title does not add a missing legend.
square	Logical: Keep the plotting grid square.
line_width	Numeric: Curve stroke width in pixels.
fold_opacity	Numeric: Opacity of individual resample curves.
xlab	Character: Horizontal axis label.
ylab	Character: Vertical axis label.
title	Optional Character: Chart title.
dat_path	Optional Character: Path to the data, read at draw time. The serializable alternative to passing data to draw().
origin	Optional Named character: Per-property provenance.
writer	Optional Named character: Interface name and version.

Value

A ROCConfig object.

Examples

```
setup_ROCConfig(class_label = "class", split = "split", fold = "fold", auc = "auc")
```

`setup_SankeyConfig` *Set up a Sankey Diagram Configuration*

Description

The seam between convenient input and a complete, validated object. **Every argument is optional**, which is what lets the published schema require nothing.

Usage

```
setup_SankeyConfig(
  source = "source",
  target = "target",
  value = "value",
  orient = "horizontal",
  node_align = NULL,
  node_width = NULL,
  node_gap = NULL,
  palette = NULL,
  title = NULL,
  dat_path = NULL,
  origin = NULL,
  writer = NULL
)
```

Arguments

`source`	Character: Column naming each link's source node.
`target`	Character: Column naming each link's target node.
`value`	Character: Column holding each link's magnitude.
`orient`	Character {"horizontal", "vertical"}: Layout direction.
`node_align`	Optional Character {"left", "right", "justify"}: How nodes are aligned along the flow.
`node_width`	Optional Numeric `[0, Inf)`: Node thickness in pixels.
`node_gap`	Optional Numeric `[0, Inf)`: Gap between nodes in pixels.
`palette`	Optional Character: Node colors, overriding the theme palette for this chart. `NULL` uses the theme's.
`title`	Optional Character: Chart title. Declared here because it is the one property every chart type has; `palette` is not (four take a `colormap` instead), and axis labels and margins are cartesian-only.

dat_path	Optional Character: Path to the data, read at draw time. The serializable alternative to passing data to draw().
origin	Optional Named character: Where each value came from. Normally computed from which arguments were supplied; pass it only when restoring a config that already carries provenance.
writer	Optional Named character: Which interface wrote the config.

Value

SankeyConfig object.

Author(s)

EDG

Examples

```
links <- data.frame(
  source = c("a", "b"),
  target = c("c", "c"),
  value = c(1, 2)
)
draw(setup_SankeyConfig(), data = links)
```

---

setup_Scatter3DConfig *Set Up a Three-Dimensional Scatter Configuration*

Description

Set Up a Three-Dimensional Scatter Configuration

Usage

```
setup_Scatter3DConfig(
  x = NULL,
  y = NULL,
  z = NULL,
  group = NULL,
  alpha = 20,
  beta = 40,
  view_size = 190,
  point_size = 8,
  opacity = NULL,
  xlab = NULL,
  ylab = NULL,
  zlab = NULL,
  palette = NULL,
```

```
  title = NULL,
  dat_path = NULL,
  origin = NULL,
  writer = NULL,
  mode = "points",
  order = "input",
  line_width = 2,
  fit = NULL,
  fit_params = NULL,
  n_fit = 25L,
  fit_alpha = 0.35,
  projection = "perspective",
  distance = 250
)
```

Arguments

x, y, z	Optional Character: Numeric coordinate column bindings.
group	Optional Character: Group column binding.
alpha	Numeric [-80, 80]: Camera elevation in degrees.
beta	Numeric [-180, 180]: Camera azimuth in degrees.
view_size	Numeric [175, 400]: Orthographic viewing extent for a 100-unit cube.
point_size	Numeric (0, 100]: Marker diameter in pixels.
opacity	Optional Numeric [0, 1]: Point and line opacity. NULL sets it from the number of complete observations, as in draw_scatter()'s point_alpha, and draws paths alone (mode = "lines") opaque.
xlab, ylab, zlab	Optional Character: Axis names.
palette	Optional Character: Group colors.
title	Optional Character: Chart title. Declared here because it is the one property every chart type has; palette is not (four take a colormap instead), and axis labels and margins are cartesian-only.
dat_path	Optional Character: Path to the data, read at draw time. The serializable alternative to passing data to draw().
origin	Optional Named character {"user", "default", "derived"}: Where each value came from, one entry per settable property. Absent on an authored config; written by the interface that resolved it.
writer	Optional Named character: Which interface wrote the config, as name and version. Absent on an authored config.
mode	Character c("points", "lines", "both"): Geometry to draw.
order	Character c("input", "x"): Order within complete path runs. Missing coordinates break paths; sorting never joins across those breaks.
line_width	Numeric (0, 100]: Path width in pixels.

fit	Optional Character: Model of z on x and y drawn as a surface per group: "glm", "gam", or an rtemis supervised learning algorithm name such as "LINAD", which requires the rtemis package (>= 1.4.1). See draw_scatter(). NULL draws no surface.
fit_params	Optional Named list: Arguments passed to the learner named in fit. See draw_scatter().
n_fit	Integer [2, Inf): Grid points per axis for fitted surfaces.
fit_alpha	Numeric [0, 1]: Opacity of fitted surfaces.
projection	Character {"perspective", "orthographic"}: Camera projection.
distance	Numeric [100, 400]: Perspective camera distance from the center of the 100-unit cube. Narrow and portrait layouts increase the distance to preserve space for axes.

Value

Scatter3DConfig: Validated settings.

Examples

```
draw(setup_Scatter3DConfig(x = "Sepal.Length", y = "Sepal.Width",
  z = "Petal.Length", group = "Species"), data = iris)
```

---

setup_ScatterConfig *Set up a Scatter Chart Configuration*

Description

The seam between convenient input and a complete, validated object: pass the handful of things you care about, get back a ScatterConfig with everything else at its default.

Usage

```
setup_ScatterConfig(
  x = NULL,
  y = NULL,
  size = NULL,
  group = NULL,
  fit = NULL,
  fit_params = NULL,
  se = TRUE,
  n_fit = 200L,
  fit_alpha = 0.25,
  palette = NULL,
  pad = DEFAULT_PAD,
  square = FALSE,
  equal_axes = FALSE,
```

```
  xlim = NULL,
  ylim = NULL,
  xlab = NULL,
  ylab = NULL,
  title = NULL,
  margin_top = NULL,
  margin_right = NULL,
  margin_bottom = NULL,
  margin_left = NULL,
  dat_path = NULL,
  origin = NULL,
  writer = NULL,
  se_times = 1.96,
  rsq = FALSE,
  diagonal = FALSE,
  diagonal_color = NULL,
  legend_position = "top",
  legend_placement = "outside",
  legend_title = NULL,
  fit_name = NULL,
  rug = FALSE,
  hover = NULL,
  point_alpha = NULL,
  zoom = FALSE
)
```

Arguments

x, y	Optional Character: Columns drawn on each axis.
size	Optional Character: Column giving per-point size.
group	Optional Character: Column to group and color points by.
fit	Optional Character: Fit to overlay: "glm", "gam", or an rtemis supervised learning algorithm name such as "LINAD", which requires the rtemis package. NULL draws no fit. See draw_scatter().
fit_params	Optional Named list: Arguments passed to the learner named in fit, e.g. list(max_leaves = 8) for "LINAD". See draw_scatter().
se	Logical: If TRUE, shade the fit standard-error band.
n_fit	Integer [2, Inf): Points used to draw the fit line.
fit_alpha	Numeric [0, 1]: Opacity of the standard-error band.
palette	Optional Character: Series colors, overriding the theme palette for this chart. NULL uses the theme's.
pad	Numeric [0, Inf): Fraction of the data range to extend each axis by when xlim / ylim are not given. The default matches base R's xaxs = "r", which extends the range by 4% at each end.
square	Logical: If TRUE, draw the plotting box square – equal height and width in pixels, excluding axis labels and margins.

`equal_axes` Logical: If TRUE, give one data unit the same size in pixels on both axes. Set with `square` for a plot that is both, such as a true-versus-predicted plot whose identity line runs at 45 degrees; the two axes are then made to span the same interval.

`xlim`, `ylim` Optional Numeric: Axis limits, length 2. `NULL` derives them from the data, padded by `pad`.

`xlab`, `ylab` Optional Character: Axis labels. `NULL` derives them from the data.

`title` Optional Character: Chart title. Declared here because it is the one property every chart type has; `palette` is not (four take a `colormap` instead), and axis labels and margins are cartesian-only.

`margin_top`, `margin_right`, `margin_bottom`, `margin_left`
Optional Integer `[0, Inf)`: Plot margins in pixels.

`dat_path` Optional Character: Path to the data, read at draw time. The serializable alternative to passing `data` to `draw()`.

`origin` Optional Named character: Where each value came from. Normally computed from which arguments were supplied; pass it only when restoring a config that already carries provenance.

`writer` Optional Named character: Which interface wrote the config, as `name` and `version`.

`se_times` Numeric `[0, Inf)`: Multiplier for the fitted standard error.

`rsq` Logical: Include the fitted model's R-squared in series labels.

`diagonal` Logical: Draw the identity line within the axis limits.

`diagonal_color` Optional Character: Identity-line color.

`legend_position`
Character {"top", "bottom", "left", "right", "top-left", "top-right", "bottom-left", "bottom-right"}: Legend anchor. Top/bottom anchors use horizontal rows; left/right anchors use a vertical column. Corner anchors align within the top or bottom row.

`legend_placement`
Character {"outside", "inside"}: Relation to the plotting area. Outside placement reserves space for the complete legend; inside placement overlays the data. Neither setting adds a missing legend.

`legend_title` Optional Character: Title drawn with the legend or colorbar: above left/right legends and vertical colorbars, before the first row of top/bottom legends and horizontal colorbars. `NULL` draws no title. A title does not add a missing legend.

`fit_name` Optional Character: Label for fitted layers.

`rug` Logical: Show marginal marks along the x and y axes.

`hover` Optional Character: Column containing per-observation tooltip labels.

`point_alpha` Optional Numeric `[0, 1]`: Point opacity. `NULL` sets it from the number of points drawn, falling from 0.9 for a handful of points to 0.15 for 100,000 or more, so that overlapping points stay visible.

`zoom` Logical: Enable Shift+wheel zoom, drag panning, and box-zoom controls on both axes. With `square` or `equal_axes`, the controls omit box zoom, which sets each axis independently.

Details

Every argument is optional, which is what lets the published schema require nothing: an authored config is a subset of the full set, and the interface fills in the rest. This is a different entry point from `draw_scatter()`, which takes vectors and keeps its mandatory `x` and `y`.

Value

ScatterConfig object.

Author(s)

EDG

Examples

```
cfg <- setup_ScatterConfig(x = "wt", y = "mpg", fit = "glm", title = "Cars")
draw(cfg, data = mtcars)
```

---

setup_SignificanceConfig

Set Up a Significance Plot Configuration

Description

Build a validated configuration with provenance for authored settings.

Usage

```
setup_SignificanceConfig(
  view = "volcano",
  estimate = "estimate",
  p_value = "p_value",
  label = NULL,
  p_adjust_method = "holm",
  n_tests = NULL,
  p_transform = "neg_log10",
  p_thresh = 0.05,
  x_thresh = 0,
  zero_cap = NULL,
  annotate_n = 7L,
  reference = TRUE,
  legend = TRUE,
  negative_color = NULL,
  neutral_color = NULL,
  positive_color = NULL,
  alpha = 0.8,
  point_size = 8,
```

```
  xlab = NULL,
  ylab = NULL,
  xlim = NULL,
  ylim = NULL,
  margin_top = NULL,
  margin_right = NULL,
  margin_bottom = NULL,
  margin_left = NULL,
  title = NULL,
  dat_path = NULL,
  origin = NULL,
  writer = NULL,
  legend_position = "top",
  legend_placement = "outside",
  legend_title = NULL,
  group = NULL,
  palette = NULL,
  zoom = FALSE
)
```

Arguments

view	Character {"volcano", "manhattan"}: Scatter or categorical bar view.
estimate	Character: Column containing coefficients or other signed effects.
p_value	Character: Column containing unadjusted p-values.
label	Optional Character: Outcome-label column. Unset generates row labels.
p_adjust_method	Character: Method from stats::p.adjust.methods. Adjustment precedes transformation.
n_tests	Optional Integer [1, Inf): Hypothesis-family size, at least the number of supplied rows. Unset counts all rows, including missing p-values.
p_transform	Character {"neg_log10", "identity", "one_minus"}: Named display transform of adjusted p-values.
p_thresh	Numeric (0, 1]: Significance threshold on adjusted p-values; the comparison is strict.
x_thresh	Numeric: Effect threshold separating low and high significant results.
zero_cap	Optional Numeric (0, Inf): Finite display height for zero p-values under neg_log10; must exceed all finite heights and the significance reference. Unset computes a cap one unit above their rounded-up maximum.
annotate_n	Integer [0, Inf): Maximum significant feature labels per side in the volcano view, ranked by raw p-value with input-order ties.
reference	Logical: Show the transformed significance threshold and, for volcanoes, the effect threshold.
legend	Logical: Show the nonempty significance groups.

`negative_color` Optional Character: Color for significant effects below x_thresh. `NULL` uses the theme's negative color.

`neutral_color` Optional Character: Color for other results, including effects equal to x_thresh. `NULL` uses the theme's neutral color.

`positive_color` Optional Character: Color for significant effects above x_thresh. `NULL` uses the theme's positive color.

`alpha` Numeric `[0, 1]`: Mark opacity.

`point_size` Numeric (0, Inf): Scatter symbol size in pixels.

`xlab` Optional Character: X-axis label.

`ylab` Optional Character: Y-axis label.

`xlim` Optional Numeric: Exactly two increasing finite volcano x-axis limits. Not used by the categorical view.

`ylim` Optional Numeric: Exactly two increasing finite y-axis limits.

`margin_top` Optional Integer [0, Inf): Top plot margin in pixels.

`margin_right` Optional Integer [0, Inf): Right plot margin in pixels.

`margin_bottom` Optional Integer [0, Inf): Bottom plot margin in pixels.

`margin_left` Optional Integer [0, Inf): Left plot margin in pixels.

`title` Optional Character: Chart title. Declared here because it is the one property every chart type has; `palette` is not (four take a `colormap` instead), and axis labels and margins are cartesian-only.

`dat_path` Optional Character: Path to the data, read at draw time. The serializable alternative to passing `data` to `draw()`.

`origin` Optional Named character: Per-property provenance.

`writer` Optional Named character: Interface name and version.

`legend_position` Character {"top", "bottom", "left", "right", "top-left", "top-right", "bottom-left", "bottom-right"}: Legend anchor. Top/bottom anchors use horizontal rows; left/right anchors use a vertical column. Corner anchors align within the top or bottom row.

`legend_placement` Character {"outside", "inside"}: Relation to the plotting area. Outside placement reserves space for the complete legend; inside placement overlays the data. Neither setting adds a missing legend.

`legend_title` Optional Character: Title drawn with the legend or colorbar: above left/right legends and vertical colorbars, before the first row of top/bottom legends and horizontal colorbars. `NULL` draws no title. A title does not add a missing legend.

`group` Optional Character: Display-group column. Unset colors by significance and effect direction. Missing groups omit marks after full-family adjustment. Groups never change significance thresholds or annotation ranking.

`palette` Optional Character vector: Display-group colors in first-appearance order; unset uses the theme. Requires a group binding.

`zoom` Logical: Enable Shift+wheel zoom and drag panning: on both axes with box-zoom, undo, and reset controls for volcano plots, and with a slider on the outcome axis for Manhattan plots.

Value

A SignificanceConfig object.

Examples

```
setup_SignificanceConfig(view = "manhattan", p_adjust_method = "BH")
```

---

setup_SpectrogramConfig

Set up a Spectrogram Configuration

Description

The seam between convenient input and a complete, validated object. **Every argument is optional**, which is what lets the published schema require nothing.

Usage

```
setup_SpectrogramConfig(
  sample_rate = NULL,
  n_fft = 256L,
  window = "hanning",
  overlap = NULL,
  power = TRUE,
  db = TRUE,
  db_range = 80,
  freq_scale = "linear",
  freq_range = NULL,
  time_range = NULL,
  freq_unit = "Hz",
  time_unit = "s",
  colormap = "magma",
  colormap_reverse = FALSE,
  n_colors = 256L,
  zlim = NULL,
  show_colorbar = TRUE,
  colorbar_title = NULL,
  xlab = NULL,
  ylab = NULL,
  title = NULL,
  margin_top = NULL,
  margin_right = NULL,
  margin_bottom = NULL,
  margin_left = NULL,
  dat_path = NULL,
  origin = NULL,
  writer = NULL,
```

```
  legend_position = "right",
  legend_placement = "outside",
  legend_title = NULL
)
```

Arguments

`sample_rate`	Optional Numeric `[0, Inf)`: Signal sample rate in Hz.
`n_fft`	Integer `[2, Inf)`: FFT window length.
`window`	Character: Window function.
`overlap`	Optional Numeric `[0, 1)`: Fractional window overlap.
`power`	Logical: Use the power spectrum rather than magnitude.
`db`	Logical: Convert to decibels.
`db_range`	Numeric `[0, Inf)`: Dynamic range in dB below the peak.
`freq_scale`	Character {"linear", "log", "mel"}: Frequency axis scale.
`freq_range`, `time_range`	Optional Numeric: Axis limits, length 2.
`freq_unit`, `time_unit`	Character: Axis units, for the labels.
`colormap`	Character: Continuous color scheme, or two or more hex colors defining one.
`colormap_reverse`	Logical: Reverse the color scheme.
`n_colors`	Integer `[2, Inf)`: Number of discrete colors.
`zlim`	Optional Numeric: Color-scale limits, length 2.
`show_colorbar`	Logical: Draw the color bar.
`colorbar_title`	Optional Character: Color bar title.
`xlab`, `ylab`	Optional Character: Axis labels.
`title`	Optional Character: Chart title. Declared here because it is the one property every chart type has; `palette` is not (four take a `colormap` instead), and axis labels and margins are cartesian-only.
`margin_top`, `margin_right`, `margin_bottom`, `margin_left`	Optional Integer `[0, Inf)`: Override the auto-computed margins, per side.
`dat_path`	Optional Character: Path to the data, read at draw time. The serializable alternative to passing `data` to `draw()`.
`origin`	Optional Named character: Where each value came from. Normally computed from which arguments were supplied; pass it only when restoring a config that already carries provenance.
`writer`	Optional Named character: Which interface wrote the config.
`legend_position`	Character {"top", "bottom", "left", "right", "top-left", "top-right", "bottom-left", "bottom-right"}: Legend anchor. Top/bottom anchors use horizontal rows; left/right anchors use a vertical column. Corner anchors align within the top or bottom row.

`legend_placement`
: Character {"outside", "inside"}: Relation to the plotting area. Outside placement reserves space for the complete legend; inside placement overlays the data. Neither setting adds a missing legend.

`legend_title`
: Optional Character: Title drawn with the legend or colorbar: above left/right legends and vertical colorbars, before the first row of top/bottom legends and horizontal colorbars. `NULL` draws no title. A title does not add a missing legend.

Value

SpectrogramConfig object.

Author(s)

EDG

Examples

```
m <- matrix(abs(rnorm(64 * 20)) + 1, nrow = 64)
draw(setup_SpectrogramConfig(), data = m)
```

---

setup_SurvivalConfig *Set Up a Survival Curve Configuration*

Description

Set Up a Survival Curve Configuration

Usage

```
setup_SurvivalConfig(
  time = "time",
  survival = "survival",
  group = NULL,
  lower = NULL,
  upper = NULL,
  n_censor = NULL,
  n_risk = NULL,
  show_ci = TRUE,
  show_censors = TRUE,
  show_median = FALSE,
  landmarks = NULL,
  risk_table = FALSE,
  ci_opacity = 0.15,
  line_width = 2,
  censor_size = 8,
  digits = 3L,
  palette = NULL,
```

```
  legend = TRUE,
  xlab = "Time",
  ylab = "Survival probability",
  legend_position = "top",
  legend_placement = "outside",
  legend_title = NULL,
  title = NULL,
  dat_path = NULL,
  origin = NULL,
  writer = NULL
)
```

Arguments

`time`	Character: Time column in the curve records.
`survival`	Character: Survival probability column.
`group`	Optional Character: Curve group column.
`lower`, `upper`	Optional Character: Pointwise confidence bound columns.
`n_censor`	Optional Character: Censor count column.
`n_risk`	Optional Character: Risk count column at recorded times.
`show_ci`	Logical: Draw available pointwise confidence bands.
`show_censors`	Logical: Draw censor ticks.
`show_median`	Logical: Draw median survival droplines when estimable.
`landmarks`	Optional Numeric vector: Finite times for survival labels.
`risk_table`	Logical: Draw supplied risk records beneath the curves.
`ci_opacity`	Numeric `[0, 1]`: Confidence band opacity.
`line_width`	Numeric: Curve stroke width in pixels.
`censor_size`	Numeric `(0, Inf)`: Censor tick height in pixels.
`digits`	Integer: Decimal places for AUC labels and tooltip values.
`palette`	Optional Character: Group colors; unset uses the chart theme.
`legend`	Logical: Show group labels and AUC summaries.
`xlab`	Character: Horizontal axis label.
`ylab`	Character: Vertical axis label.
`legend_position`	Character {"top", "bottom", "left", "right", "top-left", "top-right", "bottom-left", "bottom-right"}: Legend anchor. Top/bottom anchors use horizontal rows; left/right anchors use a vertical column. Corner anchors align within the top or bottom row.
`legend_placement`	Character {"outside", "inside"}: Relation to the plotting area. Outside placement reserves space for the complete legend; inside placement overlays the data. Neither setting adds a missing legend.

legend_title	Optional Character: Title drawn with the legend or colorbar: above left/right legends and vertical colorbars, before the first row of top/bottom legends and horizontal colorbars. NULL draws no title. A title does not add a missing legend.
title	Optional Character: Chart title.
dat_path	Optional Character: Path to the data, read at draw time. The serializable alternative to passing data to draw().
origin	Optional Named character: Per-property provenance.
writer	Optional Named character: Interface name and version.

Value

A SurvivalConfig object.

Examples

```
setup_SurvivalConfig(group = "treatment", show_median = TRUE)
```

---

setup_TimeSeriesConfig

Set Up a Time Series Configuration

Description

Set Up a Time Series Configuration

Usage

```
setup_TimeSeriesConfig(
  x = NULL,
  y = NULL,
  group = NULL,
  x2 = NULL,
  y2 = NULL,
  window = 7L,
  roll_fn = "none",
  align = "center",
  roll_width = 2,
  roll_color = NULL,
  raw_opacity = 0.5,
  points = TRUE,
  xlab = NULL,
  ylab = NULL,
  y2lab = NULL,
  palette = NULL,
  zoom = FALSE,
  title = NULL,
```

```
  dat_path = NULL,
  origin = NULL,
  writer = NULL,
  legend_position = "top",
  legend_placement = "outside",
  legend_title = NULL,
  zt = NULL,
  shade_bin = NULL,
  show_zt_every = 1L
)
```

Arguments

x, y	Optional Character: Time and value column bindings. Multiple y columns share x; group splits a single y column.
group	Optional Character: Group column binding.
x2, y2	Optional Character: Secondary time/value bindings. Unset x2 uses x.
window	Integer [1, Inf): Number of observations per rolling window.
roll_fn	Character {"none", "mean", "median", "min", "max", "sum"}: Summary.
align	Character {"center", "left", "right"}: Window alignment.
roll_width	Numeric (0, Inf): Rolling line width in pixels.
roll_color	Optional Character: Rolling line color; unset uses the group color.
raw_opacity	Numeric [0, 1]: Raw observation opacity.
points	Logical: Show raw points; FALSE draws raw lines.
xlab, ylab	Optional Character: Axis labels. NULL derives them from the data.
y2lab	Optional Character: Secondary axis name.
palette	Optional Character: Series colors, overriding the theme palette for this chart. NULL uses the theme's.
zoom	Logical: Enable an x-axis slider, Shift+wheel zoom, and drag panning.
title	Optional Character: Chart title. Declared here because it is the one property every chart type has; palette is not (four take a colormap instead), and axis labels and margins are cartesian-only.
dat_path	Optional Character: Path to the data, read at draw time. The serializable alternative to passing data to draw().
origin	Optional Named character {"user", "default", "derived"}: Where each value came from, one entry per settable property. Absent on an authored config; written by the interface that resolved it.
writer	Optional Named character: Which interface wrote the config, as name and version. Absent on an authored config.
legend_position	Character {"top", "bottom", "left", "right", "top-left", "top-right", "bottom-left", "bottom-right"}: Legend anchor. Top/bottom anchors use horizontal rows; left/right anchors use a vertical column. Corner anchors align within the top or bottom row.

`legend_placement`
: Character {"outside", "inside"}: Relation to the plotting area. Outside placement reserves space for the complete legend; inside placement overlays the data. Neither setting adds a missing legend.

`legend_title`
: Optional Character: Title drawn with the legend or colorbar: above left/right legends and vertical colorbars, before the first row of top/bottom legends and horizontal colorbars. `NULL` draws no title. A title does not add a missing legend.

`zt`, `shade_bin`
: Optional Character: Columns of zeitgeber labels and zero/one shading flags.

`show_zt_every`
: Integer `[1, Inf)`: Zeitgeber label stride; phase zero remains visible.

Value

TimeSeriesConfig: Validated settings.

Examples

```
draw(setup_TimeSeriesConfig(x = "Time", y = "demand"), data = BOD)
```

`SigmaOption` *Sigma.js Render Option*

Description

The complete, validated render spec for a Sigma.js network: a GraphModel (the data) plus all visual styling and an optional title. This is the Sigma analog of EChartsOption – the single object `draw()` dispatches on to emit a `rtemis-graph` widget. Theme is *not* a property here; like every backend, theming is supplied to `draw()` and resolved uniformly.

Usage

```
SigmaOption(
  model = NULL,
  layout = "force",
  node_size = 10,
  edge_scale = 3,
  node_opacity = 0.95,
  edge_opacity = 0.4,
  show_labels = TRUE,
  scale_by_degree = TRUE,
  color_by_group = FALSE,
  resolution = 1,
  blend_edges = FALSE,
  palette = as.character(rtemis_colors),
  node_color = rtemis_colors[["teal"]],
  positive_color = "#C55E28",
  negative_color = "#2D83D3",
  title = NULL
)
```

Arguments

model	GraphModel or named list: The graph to render (a list must contain a nodes element).
layout	Character {"force", "circular", "circlepack", "random"}: Layout algorithm. "force" is ForceAtlas2.
node_size	Numeric [0, Inf): Base node radius in screen pixels.
edge_scale	Numeric [0, Inf): Multiplier mapping normalized edge weight to stroke width.
node_opacity	Numeric [0, 1]: Node fill opacity.
edge_opacity	Numeric [0, 1]: Edge stroke opacity.
show_labels	Logical: Whether to render node labels.
scale_by_degree	Logical: Scale each node's radius by its value (weighted degree); when FALSE all nodes use the base size.
color_by_group	Logical: Color nodes by detected community (Louvain) instead of a single hue.
resolution	Numeric [0, Inf): Louvain resolution; higher yields more, smaller communities.
blend_edges	Logical: Color each edge as the blend of its two endpoint node colors instead of by sign.
palette	Character: Categorical colors for communities.
node_color	Character: Single node color used when color_by_group is FALSE.
positive_color	Character: Edge color for positive-sign edges.
negative_color	Character: Edge color for negative-sign edges.
title	Optional Character: Title shown above the network.

Details

Most users never touch this directly – draw_network() and draw_graph() build it – but power users can construct it for full control: draw(SigmaOption(model = GraphModel(...), layout = "circular")).

Its to_list() produces the { model, style, title } payload consumed by the rtemis-graph htmlwidget binding.

Value

SigmaOption object.

Examples

```
model <- GraphModel(
  nodes = list(GraphNode(id = "a"), GraphNode(id = "b")),
  edges = list(GraphEdge(source = "a", target = "b")),
  directed = FALSE
)
SigmaOption(model = model, layout = "force")
```

SignificanceConfig *Significance Plot Configuration*

Description

One portable specification for volcano plots and categorical significance bars. The latter is the MassGLM Manhattan view, not a genomic-position plot. Compiles to `ScatterSeriesOption` / `BarSeriesOption` in ECharts `src/chart/scatter/ScatterSeries.ts` and `src/chart/bar/BarSeries.ts`, with `MarkLineOption` from `src/component/marker/MarkLineModel.ts`. `to_list()` serializes the semantic configuration using schema keys; `compile()` produces the ECharts option.

Usage

```
SignificanceConfig(
  dat_path = NULL,
  title = NULL,
  origin = NULL,
  writer = NULL,
  legend_position = "top",
  legend_placement = "outside",
  legend_title = NULL,
  view = "volcano",
  estimate = "estimate",
  p_value = "p_value",
  label = NULL,
  group = NULL,
  palette = NULL,
  p_adjust_method = "holm",
  n_tests = NULL,
  p_transform = "neg_log10",
  p_thresh = 0.05,
  x_thresh = 0,
  zero_cap = NULL,
  annotate_n = 7L,
  reference = TRUE,
  legend = TRUE,
  zoom = FALSE,
  negative_color = NULL,
  neutral_color = NULL,
  positive_color = NULL,
  alpha = 0.8,
  point_size = 8,
  xlab = NULL,
  ylab = NULL,
  xlim = NULL,
  ylim = NULL,
  margin_top = NULL,
```

```
  margin_right = NULL,
  margin_bottom = NULL,
  margin_left = NULL
)
```

Arguments

dat_path	Optional Character: Path to the data, read at draw time. The serializable alternative to passing data to draw().
title	Optional Character: Chart title. Declared here because it is the one property every chart type has; palette is not (four take a colormap instead), and axis labels and margins are cartesian-only.
origin	Optional Named character {"user", "default", "derived"}: Where each value came from, one entry per settable property. Absent on an authored config; written by the interface that resolved it.
writer	Optional Named character: Which interface wrote the config, as name and version. Absent on an authored config.
legend_position	Character {"top", "bottom", "left", "right", "top-left", "top-right", "bottom-left", "bottom-right"}: Legend anchor. Top/bottom anchors use horizontal rows; left/right anchors use a vertical column. Corner anchors align within the top or bottom row.
legend_placement	Character {"outside", "inside"}: Relation to the plotting area. Outside placement reserves space for the complete legend; inside placement overlays the data. Neither setting adds a missing legend.
legend_title	Optional Character: Title drawn with the legend or colorbar: above left/right legends and vertical colorbars, before the first row of top/bottom legends and horizontal colorbars. NULL draws no title. A title does not add a missing legend.
view	Character {"volcano", "manhattan"}: Scatter or categorical bar view.
estimate	Character: Column containing coefficients or other signed effects.
p_value	Character: Column containing unadjusted p-values.
label	Optional Character: Outcome-label column. Unset generates row labels.
group	Optional Character: Display-group column. Unset colors by significance and effect direction. Missing groups omit marks after full-family adjustment. Groups never change significance thresholds or annotation ranking.
palette	Optional Character vector: Display-group colors in first-appearance order; unset uses the theme. Requires a group binding.
p_adjust_method	Character: Method from stats::p.adjust.methods. Adjustment precedes transformation.
n_tests	Optional Integer [1, Inf): Hypothesis-family size, at least the number of supplied rows. Unset counts all rows, including missing p-values.
p_transform	Character {"neg_log10", "identity", "one_minus"}: Named display transform of adjusted p-values.

p_thresh	Numeric (0, 1]: Significance threshold on adjusted p-values; the comparison is strict.
x_thresh	Numeric: Effect threshold separating low and high significant results.
zero_cap	Optional Numeric (0, Inf): Finite display height for zero p-values under neg_log10; must exceed all finite heights and the significance reference. Unset computes a cap one unit above their rounded-up maximum.
annotate_n	Integer [0, Inf): Maximum significant feature labels per side in the volcano view, ranked by raw p-value with input-order ties.
reference	Logical: Show the transformed significance threshold and, for volcanoes, the effect threshold.
legend	Logical: Show the nonempty significance groups.
zoom	Logical: Enable Shift+wheel zoom and drag panning: on both axes with box-zoom, undo, and reset controls for volcano plots, and with a slider on the outcome axis for Manhattan plots.
negative_color	Optional Character: Color for significant effects below x_thresh. NULL uses the theme's negative color.
neutral_color	Optional Character: Color for other results, including effects equal to x_thresh. NULL uses the theme's neutral color.
positive_color	Optional Character: Color for significant effects above x_thresh. NULL uses the theme's positive color.
alpha	Numeric [0, 1]: Mark opacity.
point_size	Numeric (0, Inf): Scatter symbol size in pixels.
xlab	Optional Character: X-axis label.
ylab	Optional Character: Y-axis label.
xlim	Optional Numeric: Exactly two increasing finite volcano x-axis limits. Not used by the categorical view.
ylim	Optional Numeric: Exactly two increasing finite y-axis limits.
margin_top	Optional Integer [0, Inf): Top plot margin in pixels.
margin_right	Optional Integer [0, Inf): Right plot margin in pixels.
margin_bottom	Optional Integer [0, Inf): Bottom plot margin in pixels.
margin_left	Optional Integer [0, Inf): Left plot margin in pixels.

Value

A SignificanceConfig object.

Statistical semantics

P-values are adjusted once over the full supplied family before rows with missing estimates or p-values are omitted from the marks. Missing hypotheses still count in n_tests; they do not become observed p-values of one. Raw and adjusted values remain in the compiled data. Function-valued transforms are not accepted by this portable configuration.

Under neg_log10, zero p-values have infinite mathematical height. They retain their zero value but are displayed at zero_cap, identified by triangles and a caption. No positive p-value is clamped. For identity and one-minus transforms, zeros are finite and no cap is used.

Examples

```
cfg <- setup_SignificanceConfig(estimate = "effect", p_value = "p")
draw(cfg, data = data.frame(effect = c(-2, 0.1, 3), p = c(0.001, 0.6, 0.002)))
```

`SpectrogramConfig` *Spectrogram Configuration*

Description

A serializable description of a spectrogram. Build one with `setup_SpectrogramConfig()` rather than calling this constructor directly.

Usage

```
SpectrogramConfig(
  dat_path = NULL,
  title = NULL,
  origin = NULL,
  writer = NULL,
  legend_position = "right",
  legend_placement = "outside",
  legend_title = NULL,
  sample_rate = NULL,
  n_fft = 256L,
  window = "hanning",
  overlap = NULL,
  power = TRUE,
  db = TRUE,
  db_range = 80,
  freq_scale = "linear",
  freq_range = NULL,
  time_range = NULL,
  freq_unit = "Hz",
  time_unit = "s",
  colormap = "magma",
  colormap_reverse = FALSE,
  n_colors = 256L,
  zlim = NULL,
  show_colorbar = TRUE,
  colorbar_title = NULL,
  xlab = NULL,
  ylab = NULL,
  margin_top = NULL,
  margin_right = NULL,
  margin_bottom = NULL,
  margin_left = NULL
)
```

Arguments

dat_path	Optional Character: Path to the data, read at draw time. The serializable alternative to passing data to draw().
title	Optional Character: Chart title. Declared here because it is the one property every chart type has; palette is not (four take a colormap instead), and axis labels and margins are cartesian-only.
origin	Optional Named character {"user", "default", "derived"}: Where each value came from, one entry per settable property. Absent on an authored config; written by the interface that resolved it.
writer	Optional Named character: Which interface wrote the config, as name and version. Absent on an authored config.
legend_position	Character {"top", "bottom", "left", "right", "top-left", "top-right", "bottom-left", "bottom-right"}: Legend anchor. Top/bottom anchors use horizontal rows; left/right anchors use a vertical column. Corner anchors align within the top or bottom row.
legend_placement	Character {"outside", "inside"}: Relation to the plotting area. Outside placement reserves space for the complete legend; inside placement overlays the data. Neither setting adds a missing legend.
legend_title	Optional Character: Title drawn with the legend or colorbar: above left/right legends and vertical colorbars, before the first row of top/bottom legends and horizontal colorbars. NULL draws no title. A title does not add a missing legend.
sample_rate	Optional Numeric [0, Inf): Signal sample rate in Hz.
n_fft	Integer [2, Inf): FFT window length.
window	Character: Window function.
overlap	Optional Numeric [0, 1): Fractional window overlap.
power	Logical: Use the power spectrum rather than magnitude.
db	Logical: Convert to decibels.
db_range	Numeric [0, Inf): Dynamic range in dB below the peak.
freq_scale	Character {"linear", "log", "mel"}: Frequency axis scale.
freq_range, time_range	Optional Numeric: Axis limits, length 2.
freq_unit, time_unit	Character: Axis units, for the labels.
colormap	Character: Continuous color scheme, or two or more hex colors defining one.
colormap_reverse	Logical: Reverse the color scheme.
n_colors	Integer [2, Inf): Number of discrete colors.
zlim	Optional Numeric: Color-scale limits, length 2.
show_colorbar	Logical: Draw the color bar.
colorbar_title	Optional Character: Color bar title.
xlab, ylab	Optional Character: Axis labels.
margin_top, margin_right, margin_bottom, margin_left	Optional Integer [0, Inf): Override the auto-computed margins, per side.

Details

The bound data is either a numeric signal vector, which is transformed with a short-time Fourier transform, or a precomputed frequency-by-time matrix.

Value

`SpectrogramConfig` object.

Author(s)

EDG

Examples

```
setup_SpectrogramConfig(n_fft = 512L)@type
```

`SplitArea` *Split Area*

Description

Configuration for alternating colored bands between split lines.

Usage

```
SplitArea(show = NULL, interval = NULL, area_style = NULL)
```

Arguments

`show`	Optional Logical: Whether to show the split area.
`interval`	Optional Numeric or Character {"auto"}: Split-area interval.
`area_style`	Optional AreaStyle: Area styling.

Details

Corresponds to `SplitAreaOption` in `src/coord/axisCommonTypes.ts` (line 369). ECharts docs: https://echarts.apache.org/en/option.html#xAxis.splitArea

Value

`SplitArea` object.

Examples

```
SplitArea(show = TRUE, area_style = AreaStyle(opacity = 0.1))
```

SplitLine *Split Line*

Description

Configuration for grid split lines perpendicular to the axis.

Usage

```
SplitLine(
  show = NULL,
  interval = NULL,
  show_min_line = NULL,
  show_max_line = NULL,
  line_style = NULL
)
```

Arguments

show	Optional Logical: Whether to show split lines.
interval	Optional Numeric or Character {"auto"}: Split-line interval.
show_min_line	Optional Logical: Whether to show the split line at the minimum end.
show_max_line	Optional Logical: Whether to show the split line at the maximum end.
line_style	Optional LineStyle: Line styling.

Details

Corresponds to SplitLineOption in src/coord/axisCommonTypes.ts (line 353). ECharts docs: https://echarts.apache.org/en/option.html#xAxis.splitLine

Value

SplitLine object.

Examples

```
SplitLine(show = TRUE, line_style = LineStyle(type = "dashed"))
```

summarize_varimp *Summarize and Rank Variable Importance*

Description

Summarize each variable's scores across folds and rank the variables, as `draw_varimp()` selects and orders them.

Usage

```
summarize_varimp(
  x,
  measure = NULL,
  top_n = NULL,
  rank_by = "magnitude",
  summary = "mean",
  folds = NULL,
  decreasing = TRUE
)
```

Arguments

`x`	Numeric vector, single-column matrix, or data frame: Importance scores. Tables require `variable` and named numeric measures, optionally `fold`. Variable names must be unique within each fold.
`measure`	Optional Character: Measure column. NULL selects the first column other than `variable` or `fold`; vectors use `"importance"`.
`top_n`	Optional Integer `[1, Inf)`: Maximum number of variables. NULL includes all variables with a summary; fractions are not accepted.
`rank_by`	Character {"magnitude", "signed"}: Rank by absolute or signed summary, respectively.
`summary`	Character {"mean", "median"}: Summary across folds.
`folds`	Optional Character: Full set of fold IDs, including folds with no rows. NULL uses the IDs observed in the `fold` column.
`decreasing`	Logical: Select and display ranks from largest to smallest. FALSE selects and displays the smallest ranks first.

Details

Each variable's summary is the mean or median of its available scores. Explicit NA scores, omitted rows, and wholly unavailable folds are excluded from the summary, and variables with no available score are omitted. Records without a `fold` column have one score per variable, which is its summary. Magnitude ranking uses the absolute summary, not the mean absolute fold score; signed ranking uses the summary itself. Ties retain first appearance in the input.

A producer whose importance omits variables with a zero score supplies those zeros explicitly; an omitted row is an unavailable score.

Value

Data frame with one row per selected variable, in rank order: `variable`, `importance` (the summary), `n_available` (folds with a score), and `n_folds` (all folds).

Examples

```
scores <- data.frame(
  variable = c("age", "weight", "age", "weight"),
  fold = c("A", "A", "B", "B"),
  coefficient = c(0.8, -1.2, 0.6, -0.9)
)
summarize_varimp(scores)
summarize_varimp(scores, rank_by = "signed")
```

SurfaceLayer *Supplied Three-Dimensional Surface*

Description

A materialized grid of predictions z = f(x, y), independent of the model that produced them. Corresponds to ECharts-GL `SurfaceSeries` in `src/chart/surface/SurfaceSeries.js`. The grid uses flat color shading and no wireframe, matching the rtemislive surface contract. Each grid rectangle is split along ECharts-GL's top-left to bottom-right diagonal. Missing z values remove every adjacent rectangle; they are not interpolated.

Usage

```
SurfaceLayer(
  x = c(0, 1),
  y = c(0, 1),
  z = NULL,
  name = "Surface",
  color = "#6ca3a0",
  opacity = 0.35
)
```

Arguments

x, y	Numeric: Finite, distinct grid coordinates, at least two per axis.
z	Numeric matrix: Predictions, with x along rows and y along columns. NA removes adjacent cells. At least one complete grid cell is required.
name	Character: Legend label; matching names link legend selection.
color	Character: Surface fill color.
opacity	Numeric `[0, 1]`: Surface opacity.

Details

Coordinates are sorted together with the prediction matrix. SVG export uses the configured perspective or orthographic camera and splits intersecting polygons to order visible surfaces, paths and points. Dense or highly intersecting surfaces may require a coarser prediction grid for practical vector output.

Value

SurfaceLayer: Validated prediction grid.

Examples

```
SurfaceLayer(x = 0:2, y = 0:2, z = outer(0:2, 0:2, `+`))
```

SurvivalConfig *Survival Curve Configuration*

Description

Draw precomputed survival estimates, without fitting a model in the renderer. Corresponds to LineSeriesOption in src/chart/line/LineSeries.ts and ScatterSeriesOption in src/chart/scatter/ScatterSer
ECharts docs: https://echarts.apache.org/en/option.html#series-line.step

Usage

```
SurvivalConfig(
  dat_path = NULL,
  title = NULL,
  origin = NULL,
  writer = NULL,
  legend_position = "top",
  legend_placement = "outside",
  legend_title = NULL,
  time = "time",
  survival = "survival",
  group = NULL,
  lower = NULL,
  upper = NULL,
  n_censor = NULL,
  n_risk = NULL,
  show_ci = TRUE,
  show_censors = TRUE,
  show_median = FALSE,
  landmarks = NULL,
  risk_table = FALSE,
  ci_opacity = 0.15,
  line_width = 2,
```

```
  censor_size = 8,
  digits = 3L,
  palette = NULL,
  legend = TRUE,
  xlab = "Time",
  ylab = "Survival probability"
)
```

Arguments

dat_path	Optional Character: Path to the data, read at draw time. The serializable alternative to passing data to draw().
title	Optional Character: Chart title.
origin	Optional Named character {"user", "default", "derived"}: Where each value came from, one entry per settable property. Absent on an authored config; written by the interface that resolved it.
writer	Optional Named character: Which interface wrote the config, as name and version. Absent on an authored config.
legend_position	Character {"top", "bottom", "left", "right", "top-left", "top-right", "bottom-left", "bottom-right"}: Legend anchor. Top/bottom anchors use horizontal rows; left/right anchors use a vertical column. Corner anchors align within the top or bottom row.
legend_placement	Character {"outside", "inside"}: Relation to the plotting area. Outside placement reserves space for the complete legend; inside placement overlays the data. Neither setting adds a missing legend.
legend_title	Optional Character: Title drawn with the legend or colorbar: above left/right legends and vertical colorbars, before the first row of top/bottom legends and horizontal colorbars. NULL draws no title. A title does not add a missing legend.
time	Character: Time column in the curve records.
survival	Character: Survival probability column.
group	Optional Character: Curve group column.
lower, upper	Optional Character: Pointwise confidence bound columns.
n_censor	Optional Character: Censor count column.
n_risk	Optional Character: Risk count column at recorded times.
show_ci	Logical: Draw available pointwise confidence bands.
show_censors	Logical: Draw censor ticks.
show_median	Logical: Draw median survival droplines when estimable.
landmarks	Optional Numeric vector: Finite times for survival labels.
risk_table	Logical: Draw supplied risk records beneath the curves.
ci_opacity	Numeric [0, 1]: Confidence band opacity.
line_width	Numeric: Curve stroke width in pixels.

censor_size	Numeric (0, Inf): Censor tick height in pixels.
digits	Integer: Decimal places for AUC labels and tooltip values.
palette	Optional Character: Group colors; unset uses the chart theme.
legend	Logical: Show group labels and AUC summaries.
xlab	Character: Horizontal axis label.
ylab	Character: Vertical axis label.

Value

A SurvivalConfig object.

Records and statistical semantics

Supply a curve data frame, or a list with curves and optional risk data frames. Each curve row contains time and survival probability, with optional group, pointwise lower/upper confidence bounds, censor counts, and risk counts. Include the initial time/probability explicitly: the renderer never invents an origin. Records are sorted within groups; times must be unique and finite, probabilities nonincreasing in [0, 1]. Curves are right-continuous and stop at their last recorded time. Missing confidence bounds produce gaps, not zeros. A censor tick represents one or more censored observations at that time; its tooltip retains the supplied count. Counts may be fractional for weights. Median survival is the first crossing of 0.5; a plateau at 0.5 uses its midpoint, including the last follow-up time if the plateau is terminal. Comparisons to 0.5 use tolerance sqrt(.Machine$double.eps), as in survival. Landmarks outside a group's observed time range are omitted without extrapolation. Display precision never rounds plotted coordinates.

Risk records have fixed columns time, group, and n_risk, and state the risk count immediately before the specified time. They must cover the same unique times in each group. Unknown counts use NA and display as NA. The renderer does not interpolate or infer risk sets from the curves. All layers belonging to a group share its legend toggle, including risk rows.

Examples

```
cfg <- setup_SurvivalConfig()
draw(cfg, data = data.frame(time = c(0, 1, 2), survival = c(1, .8, .5)))
```

TextStyle *Text Style*

Description

Text styling used in labels, titles, tooltips, and other text elements.

Usage

```
TextStyle(
  color = NULL,
  font_style = NULL,
  font_weight = NULL,
  font_family = NULL,
  font_size = NULL,
  align = NULL,
  vertical_align = NULL,
  opacity = NULL,
  line_height = NULL,
  background_color = NULL,
  border_color = NULL,
  border_width = NULL,
  border_type = NULL,
  border_dash_offset = NULL,
  border_radius = NULL,
  padding = NULL,
  width = NULL,
  height = NULL,
  text_border_color = NULL,
  text_border_width = NULL,
  text_border_type = NULL,
  text_border_dash_offset = NULL,
  text_shadow_blur = NULL,
  text_shadow_color = NULL,
  text_shadow_offset_x = NULL,
  text_shadow_offset_y = NULL,
  overflow = NULL,
  ellipsis = NULL,
  rich = NULL,
  shadow_blur = NULL,
  shadow_color = NULL,
  shadow_offset_x = NULL,
  shadow_offset_y = NULL
)
```

Arguments

color	Optional Character: Text color.
font_style	Optional Character {"normal", "italic", "oblique"}: Font style.
font_weight	Optional Character {"normal", "bold", "bolder", "lighter"} or Numeric [100, 900]: Font weight.
font_family	Optional Character: Font family name.
font_size	Optional Numeric or Character: Font size.
align	Optional Character {"left", "center", "right"}: Horizontal alignment.
vertical_align	Optional Character {"top", "middle", "bottom"}: Vertical alignment.

`opacity` Optional Numeric `[0, 1]`: Text opacity.

`line_height` Optional Numeric `[0, Inf)`: Line height in pixels.

`background_color` Optional Character: Background color behind text.

`border_color` Optional Character: Text box border color.

`border_width` Optional Numeric `[0, Inf)`: Text box border width.

`border_type` Optional Character {"solid", "dashed", "dotted"}: Text box border type.

`border_dash_offset` Optional Numeric: Text box border dash offset.

`border_radius` Optional Numeric or Numeric vector: Text box corner radius.

`padding` Optional Numeric or Numeric vector: Text box padding.

`width` Optional Numeric or Character: Text box width.

`height` Optional Numeric `[0, Inf)`: Text box height.

`text_border_color` Optional Character: Stroke color around the text.

`text_border_width` Optional Numeric `[0, Inf)`: Stroke width around the text.

`text_border_type` Optional Character {"solid", "dashed", "dotted"}: Stroke type around the text.

`text_border_dash_offset` Optional Numeric: Stroke dash offset around the text.

`text_shadow_blur` Optional Numeric `[0, Inf)`: Text shadow blur radius.

`text_shadow_color` Optional Character: Text shadow color.

`text_shadow_offset_x` Optional Numeric: Text shadow horizontal offset.

`text_shadow_offset_y` Optional Numeric: Text shadow vertical offset.

`overflow` Optional Character {"truncate", "break", "breakAll", "none"}: Text overflow behavior.

`ellipsis` Optional Character: Ellipsis string used when truncating.

`rich` Optional Named list: Rich text styles.

`shadow_blur` Optional Numeric `[0, Inf)`: Box shadow blur radius.

`shadow_color` Optional Character: Box shadow color.

`shadow_offset_x` Optional Numeric: Box shadow horizontal offset.

`shadow_offset_y` Optional Numeric: Box shadow vertical offset.

Details

Corresponds to `TextCommonOption` in `src/util/types.ts` (line 1220). ECharts docs: https://echarts.apache.org/en/option.html#title.textStyle

Value

`TextStyle` object.

Examples

```
TextStyle(color = "#333333", font_size = 14, font_weight = "bold")
```

`Theme` *Theme*

Description

A chart theme that controls default colors, text styles, and component appearance. Corresponds to the object passed as the second argument to `echarts.init(dom, theme)` or registered via `echarts.registerTheme()`.

Usage

```
Theme(
  color = NULL,
  background_color = NULL,
  negative_color = NULL,
  neutral_color = NULL,
  positive_color = NULL,
  text_style = NULL,
  title = NULL,
  legend = NULL,
  tooltip = NULL,
  data_zoom = NULL,
  line = NULL,
  bar = NULL,
  pie = NULL,
  scatter = NULL,
  category_axis = NULL,
  value_axis = NULL,
  log_axis = NULL,
  time_axis = NULL
)
```

Arguments

`color`	Optional Character: Color palette.
`background_color`	Optional Character: Chart background color.
`negative_color`, `neutral_color`, `positive_color`	Optional Character: Colors for negative values, values of neither sign, and positive values, in every chart that colors by sign. A chart's own color arguments take precedence; `NULL` uses the defaults of `theme_light()`.

text_style	Optional TextStyle: Global default text style.
title	Optional Named list: Title overrides.
legend	Optional Named list: Legend overrides.
tooltip	Optional Named list: Tooltip overrides.
data_zoom	Optional Named list: Data-zoom defaults using native ECharts camelCase fields from SliderDataZoomOption in src/component/dataZoom/SliderZoomModel.ts.
line	Optional Named list: Line-series defaults.
bar	Optional Named list: Bar-series defaults.
pie	Optional Named list: Pie-series defaults.
scatter	Optional Named list: Scatter-series defaults.
category_axis	Optional Named list: Category-axis defaults.
value_axis	Optional Named list: Value-axis defaults.
log_axis	Optional Named list: Log-axis defaults.
time_axis	Optional Named list: Time-axis defaults.

Details

TS source: src/model/globalDefault.ts, src/theme/dark.ts.

Value

Theme object.

Examples

```
Theme(color = c("#16a085", "#e67e22"), background_color = "#ffffff")
```

---

theme_dark *Dark Theme*

Description

Returns a dark theme. All parameters have sensible defaults; most users only need base_font_size. Title font size defaults to 1.2x the base; all other text defaults to 1x. Data-zoom sliders use neutral gray controls adapted to dark backgrounds.

Usage

```
theme_dark(
  base_font_size = 12,
  title_font_size = NULL,
  subtitle_font_size = NULL,
  axis_label_font_size = NULL,
  legend_font_size = NULL,
  tooltip_font_size = NULL,
  font_family = getOption("rtemis.font_family"),
  color = palette_colors(rtemis_colors),
  bg_color = "#181818",
  fg_color = "rgba(255, 255, 255, 0.7)",
  title_color = "rgba(255, 255, 255, 0.9)",
  subtitle_color = "rgba(255, 255, 255, 0.5)",
  legend_color = NULL,
  axis_color = "rgba(255, 255, 255, 0.3)",
  grid_color = "rgba(255, 255, 255, 0.1)",
  tooltip_bg = "rgba(20, 20, 20, 0.9)",
  tooltip_border_color = "rgba(255, 255, 255, 0.1)",
  tooltip_color = NULL,
  negative_color = SIGN_COLORS[["negative"]],
  neutral_color = SIGN_COLORS[["neutral"]],
  positive_color = SIGN_COLORS[["positive"]]
)
```

Arguments

`base_font_size` Numeric `[0, Inf)`: Base font size in pixels for all text.

`title_font_size`
: Optional Numeric `[0, Inf)`: Title font size.

`subtitle_font_size`
: Optional Numeric `[0, Inf)`: Subtitle font size.

`axis_label_font_size`
: Optional Numeric `[0, Inf)`: Axis-label font size.

`legend_font_size`
: Optional Numeric `[0, Inf)`: Legend font size.

`tooltip_font_size`
: Optional Numeric `[0, Inf)`: Tooltip font size.

`font_family` Optional Character scalar: Font family name or CSS font stack, e.g. `"Geist"` or `"Geist, Helvetica, sans-serif"`. Unless the stack ends in a generic family such as `sans-serif`, the system UI font stack follows it as the fallback. A family name that is not a sequence of CSS identifiers, e.g. `Source Sans 3`, is quoted. `NULL` uses the system UI font stack. The default is the `rtemis.font_family` option, which sets the font of every rtemis plot, e.g. `options(rtemis.font_family = "Geist")`. A widget uses the font when the viewer's system has it installed or the page loads it.

`color` Character: Color palette.

bg_color Optional Character: Chart background color.

fg_color Optional Character: Global text color.

title_color Optional Character: Title text color.

subtitle_color Optional Character: Subtitle text color.

legend_color Optional Character: Legend text color.

axis_color Optional Character: Axis line color.

grid_color Optional Character: Grid or split-line color.

tooltip_bg Optional Character: Tooltip background color.

tooltip_border_color
Optional Character: Tooltip border color.

tooltip_color Optional Character: Tooltip text color.

negative_color, neutral_color, positive_color
Character: Colors for negative values, values of neither sign, and positive values, in every chart that colors by sign. The defaults, blue, gray and orange, have equal perceived lightness, keep at least 3.9:1 contrast on light and dark backgrounds, and stay distinguishable under deuteranopia and protanopia.

Value

Theme: Theme object.

Examples

```
theme_dark(base_font_size = 14)
```

---

theme_light *Light Theme*

Description

Returns a light theme. All parameters have sensible defaults; most users only need base_font_size. Title font size defaults to 1.2x the base; all other text defaults to 1x. Data-zoom sliders use neutral gray controls, including their data preview and hover states. Explicit component styles override these theme defaults.

Usage

```
theme_light(
  base_font_size = 12,
  title_font_size = NULL,
  subtitle_font_size = NULL,
  axis_label_font_size = NULL,
  legend_font_size = NULL,
  tooltip_font_size = NULL,
  font_family = getOption("rtemis.font_family"),
```

```
  color = palette_colors(rtemis_colors),
  bg_color = "#ffffff",
  fg_color = "rgba(0, 0, 0, 0.7)",
  title_color = NULL,
  subtitle_color = NULL,
  legend_color = NULL,
  axis_color = NULL,
  grid_color = NULL,
  tooltip_bg = NULL,
  tooltip_border_color = NULL,
  tooltip_color = NULL,
  negative_color = SIGN_COLORS[["negative"]],
  neutral_color = SIGN_COLORS[["neutral"]],
  positive_color = SIGN_COLORS[["positive"]]
)
```

Arguments

base_font_size	Numeric [0, Inf): Base font size in pixels for all text.
title_font_size	Optional Numeric [0, Inf): Title font size.
subtitle_font_size	Optional Numeric [0, Inf): Subtitle font size.
axis_label_font_size	Optional Numeric [0, Inf): Axis-label font size.
legend_font_size	Optional Numeric [0, Inf): Legend font size.
tooltip_font_size	Optional Numeric [0, Inf): Tooltip font size.
font_family	Optional Character scalar: Font family name or CSS font stack, e.g. "Geist" or "Geist, Helvetica, sans-serif". Unless the stack ends in a generic family such as sans-serif, the system UI font stack follows it as the fallback. A family name that is not a sequence of CSS identifiers, e.g. Source Sans 3, is quoted. NULL uses the system UI font stack. The default is the rtemis.font_family option, which sets the font of every rtemis plot, e.g. options(rtemis.font_family = "Geist"). A widget uses the font when the viewer's system has it installed or the page loads it.
color	Character: Color palette.
bg_color	Optional Character: Chart background color.
fg_color	Optional Character: Global text color.
title_color	Optional Character: Title text color.
subtitle_color	Optional Character: Subtitle text color.
legend_color	Optional Character: Legend text color.
axis_color	Optional Character: Axis line color.
grid_color	Optional Character: Grid or split-line color.

`tooltip_bg` Optional Character: Tooltip background color.

`tooltip_border_color`
Optional Character: Tooltip border color.

`tooltip_color` Optional Character: Tooltip text color.

`negative_color`, `neutral_color`, `positive_color`
Character: Colors for negative values, values of neither sign, and positive values, in every chart that colors by sign. The defaults, blue, gray and orange, have equal perceived lightness, keep at least 3.9:1 contrast on light and dark backgrounds, and stay distinguishable under deuteranopia and protanopia.

Value

Theme: Theme object.

Examples

```
theme_light(base_font_size = 14)
```

`TimeSeriesConfig` *Time Series Configuration*

Description

Numeric or date-time observations with optional rolling summaries and a secondary value axis. Uses LineSeriesOption and CartesianAxisOption from ECharts src/chart/line and src/coord/cartesian. Rolling windows count sorted observations, not elapsed time; incomplete windows and windows containing missing values remain missing. Even centered windows favor following values, matching data.table's centered alignment used by rtemis.

Usage

```
TimeSeriesConfig(
  dat_path = NULL,
  title = NULL,
  origin = NULL,
  writer = NULL,
  legend_position = "top",
  legend_placement = "outside",
  legend_title = NULL,
  x = NULL,
  y = NULL,
  group = NULL,
  xlab = NULL,
  ylab = NULL,
  palette = NULL,
  zoom = FALSE,
  x2 = NULL,
```

```
  zt = NULL,
  shade_bin = NULL,
  show_zt_every = 1L,
  y2 = NULL,
  y2lab = NULL,
  window = 7L,
  roll_fn = "none",
  align = "center",
  roll_width = 2,
  roll_color = NULL,
  raw_opacity = 0.5,
  points = TRUE
)
```

Arguments

dat_path	Optional Character: Path to the data, read at draw time. The serializable alternative to passing data to draw().
title	Optional Character: Chart title. Declared here because it is the one property every chart type has; palette is not (four take a colormap instead), and axis labels and margins are cartesian-only.
origin	Optional Named character {"user", "default", "derived"}: Where each value came from, one entry per settable property. Absent on an authored config; written by the interface that resolved it.
writer	Optional Named character: Which interface wrote the config, as name and version. Absent on an authored config.
legend_position	Character {"top", "bottom", "left", "right", "top-left", "top-right", "bottom-left", "bottom-right"}: Legend anchor. Top/bottom anchors use horizontal rows; left/right anchors use a vertical column. Corner anchors align within the top or bottom row.
legend_placement	Character {"outside", "inside"}: Relation to the plotting area. Outside placement reserves space for the complete legend; inside placement overlays the data. Neither setting adds a missing legend.
legend_title	Optional Character: Title drawn with the legend or colorbar: above left/right legends and vertical colorbars, before the first row of top/bottom legends and horizontal colorbars. NULL draws no title. A title does not add a missing legend.
x, y	Optional Character: Time and value column bindings. Multiple y columns share x; group splits a single y column.
group	Optional Character: Group column binding.
xlab, ylab	Optional Character: Axis labels. NULL derives them from the data.
palette	Optional Character: Series colors, overriding the theme palette for this chart. NULL uses the theme's.
zoom	Logical: Enable an x-axis slider, Shift+wheel zoom, and drag panning.
x2, y2	Optional Character: Secondary time/value bindings. Unset x2 uses x.

zt, shade_bin	Optional Character: Columns of zeitgeber labels and zero/one shading flags.
show_zt_every	Integer [1, Inf): Zeitgeber label stride; phase zero remains visible.
y2lab	Optional Character: Secondary axis name.
window	Integer [1, Inf): Number of observations per rolling window.
roll_fn	Character {"none", "mean", "median", "min", "max", "sum"}: Summary.
align	Character {"center", "left", "right"}: Window alignment.
roll_width	Numeric (0, Inf): Rolling line width in pixels.
roll_color	Optional Character: Rolling line color; unset uses the group color.
raw_opacity	Numeric [0, 1]: Raw observation opacity.
points	Logical: Show raw points; FALSE draws raw lines.

Value

TimeSeriesConfig: Validated portable settings.

Examples

```
setup_TimeSeriesConfig(x = "day", y = "value", window = 3L)
```

Title *Title*

Description

Chart title configuration with main text and optional subtext.

Usage

```
Title(
  show = NULL,
  text = NULL,
  link = NULL,
  target = NULL,
  subtext = NULL,
  sublink = NULL,
  subtarget = NULL,
  text_align = NULL,
  text_vertical_align = NULL,
  padding = NULL,
  item_gap = NULL,
  text_style = NULL,
  subtext_style = NULL,
  trigger_event = NULL,
  background_color = NULL,
  border_color = NULL,
```

```
  border_width = NULL,
  border_radius = NULL,
  left = NULL,
  right = NULL,
  top = NULL,
  bottom = NULL
)
```

Arguments

show	Optional Logical: Whether to show the title.
text	Optional Character: Main title text. Supports \\n for newlines.
link	Optional Character: Hyperlink for the main title.
target	Optional Character {"self", "blank"}: Link target.
subtext	Optional Character: Subtitle text.
sublink	Optional Character: Hyperlink for the subtitle.
subtarget	Optional Character {"self", "blank"}: Subtitle link target.
text_align	Optional Character {"auto", "left", "center", "right"}: Horizontal alignment.
text_vertical_align	Optional Character {"auto", "top", "middle", "bottom"}: Vertical alignment.
padding	Optional Numeric: Padding inside the title box.
item_gap	Optional Numeric [0, Inf): Gap between title and subtitle in pixels.
text_style	Optional TextStyle: Main title text style.
subtext_style	Optional TextStyle: Subtitle text style.
trigger_event	Optional Logical: Whether the title emits events.
background_color	Optional Character: Title-box background color.
border_color	Optional Character: Border color.
border_width	Optional Numeric [0, Inf): Border width.
border_radius	Optional Numeric: Border radius.
left	Optional Numeric or Character: Left position.
right	Optional Numeric or Character: Right position.
top	Optional Numeric or Character: Top position.
bottom	Optional Numeric or Character: Bottom position.

Details

Corresponds to TitleOption in src/component/title/install.ts (line 48). ECharts docs: https://echarts.apache.org/en/option.html#title

Value

Title object.

Examples

```
Title(text = "Monthly revenue", subtext = "2026", left = "center")
```

`Tooltip` *Tooltip*

Description

Tooltip configuration for showing data on hover/click.

Usage

```
Tooltip(
  show = NULL,
  trigger = NULL,
  trigger_on = NULL,
  show_content = NULL,
  always_show_content = NULL,
  formatter = NULL,
  value_formatter = NULL,
  position = NULL,
  confine = NULL,
  enterable = NULL,
  show_delay = NULL,
  hide_delay = NULL,
  transition_duration = NULL,
  background_color = NULL,
  border_color = NULL,
  border_width = NULL,
  border_radius = NULL,
  padding = NULL,
  text_style = NULL,
  extra_css_text = NULL,
  order = NULL,
  class_name = NULL
)
```

Arguments

`show`	Optional Logical: Whether to show the tooltip.
`trigger`	Optional Character {"item", "axis", "none"}: Trigger type.
`trigger_on`	Optional Character {"mousemove", "click", "none", "mousemovelclick"}: Trigger event. `"mousemove|click"`.
`show_content`	Optional Logical: Whether to show tooltip content.
`always_show_content`	Optional Logical: Whether to keep the tooltip visible.

formatter Optional Character or function: Tooltip content formatter.

value_formatter Optional function: Formatter for the value portion.

position Optional Character or Numeric: Tooltip position.

confine Optional Logical: Whether to confine the tooltip within the chart container.

enterable Optional Logical: Whether the mouse can enter the tooltip.

show_delay Optional Numeric [0, Inf): Delay before showing, in milliseconds.

hide_delay Optional Numeric [0, Inf): Delay before hiding, in milliseconds.

transition_duration Optional Numeric [0, Inf): Transition duration in seconds.

background_color Optional Character: Background color.

border_color Optional Character: Border color.

border_width Optional Numeric [0, Inf): Border width.

border_radius Optional Numeric [0, Inf): Border radius.

padding Optional Numeric: Padding inside the tooltip.

text_style Optional TextStyle: Tooltip text style.

extra_css_text Optional Character: Extra CSS for HTML render mode.

order Optional Character {"seriesAsc", "seriesDesc", "valueAsc", "valueDesc"}: Tooltip content order.

class_name Optional Character: CSS class name for the tooltip element.

Details

Corresponds to TooltipOption in src/component/tooltip/TooltipModel.ts (line 36) and CommonTooltipOption in src/util/types.ts (line 1527). ECharts docs: https://echarts.apache.org/en/option.html#tooltip

Value

Tooltip object.

Examples

```
Tooltip(trigger = "axis", border_width = 1)
```

to_json *Convert EChartsOption to JSON*

Description

Convert EChartsOption to JSON

Usage

```
to_json(x, pretty = FALSE, auto_unbox = TRUE, ...)
```

Arguments

x	EChartsOption: Option object to serialize.
pretty	Logical: Whether to pretty-print the JSON.
auto_unbox	Logical: Whether to auto-unbox single-element vectors.
...	Dots: Additional arguments passed to `jsonlite::toJSON()`.

Value

Character: JSON string.

Examples

```
to_json(EChartsOption(title = Title(text = "Weekly sessions")))
```

to_list *Convert to echarts-compatible list*

Description

Generic function that converts an S7 object to a plain named list matching the ECharts JSON option structure.

Usage

```
to_list(x, ...)
```

Arguments

x	S7 object: Object to serialize.
...	Dots: Unused.

Value

List: Named list.

Examples

```
to_list(LineStyle(color = "#16a085", width = 2))
```

---

true_pred_data *Normalize paired regression observations*

Description

Named lists are matched by name; unnamed lists are matched by position. Each pair is checked before concatenation, so unequal split or fold lengths cannot silently shift observations. Missing pairs are removed together.

Usage

```
true_pred_data(x, y, group = NULL)
```

Arguments

x	Numeric vector, single-column matrix, or list: True observations.
y	Numeric vector, single-column matrix, or list: Predicted values.
group	Optional Atomic vector or single-column data frame: Groups for vector inputs.

Value

Data frame with `true`, `predicted`, and optional `sample` columns.

Examples

```
true_pred_data(1:3, c(1.1, 1.9, 3.2))
```

---

varimp_data *Normalize variable-importance records*

Description

Keep named measures and observed fold rows intact. Summaries are computed separately so this boundary can also supply distribution views.

Usage

```
varimp_data(x)
```

Arguments

x	Numeric vector, single-column matrix, or data frame: Importance data.

Value

Data frame with `variable`, named measures, and optionally `fold`.

Examples

```
varimp_data(c(age = 2, weight = 1))
```

`VisualMap` *Visual Map*

Description

Continuous visual map component that maps data values to visual properties (typically color). Required for heatmaps.

Usage

```
VisualMap(
  type = "continuous",
  min = NULL,
  max = NULL,
  precision = NULL,
  calculable = NULL,
  show = NULL,
  orient = NULL,
  in_range = NULL,
  out_of_range = NULL,
  left = NULL,
  right = NULL,
  top = NULL,
  bottom = NULL
)
```

Arguments

`type`	Character {"continuous", "piecewise"}: Visual map type.
`min`	Optional Numeric: Minimum value of the mapped data range.
`max`	Optional Numeric: Maximum value of the mapped data range.
`precision`	Optional Integer `[0, Inf)`: Decimal places shown in colorbar labels and the draggable-handle tooltip. ECharts defaults to `0`, which rounds labels to whole numbers — set this to match `value_digits` for accurate display.
`calculable`	Optional Logical: Whether to show drag handles for interactive range selection.
`show`	Optional Logical: Whether to display the visual map legend.
`orient`	Optional Character {"vertical", "horizontal"}: Orientation of the color bar.

in_range	Optional list: Visual encoding for in-range values. Typically list(color = c("blue", "white", "red")).
out_of_range	Optional list: Visual encoding for out-of-range values.
left	Optional Numeric or Character: Distance from the left of the container.
right	Optional Numeric or Character: Distance from the right of the container.
top	Optional Numeric or Character: Distance from the top of the container. For vertical heatmap colorbars, draw() aligns "middle" and "center" with the plotted data area.
bottom	Optional Numeric or Character: Distance from the bottom of the container.

Details

Corresponds to ContinuousVisualMapOption in src/component/visualMap/ContinuousModel.ts. ECharts docs: https://echarts.apache.org/en/option.html#visualMap-continuous

Value

VisualMap object.

Examples

```
VisualMap(type = "continuous", min = 0, max = 100, calculable = TRUE)
```

---

write_chart_config *Write a chart config to a JSON file*

Description

Write a chart config to a JSON file

Usage

```
write_chart_config(config, path, complete = FALSE)
```

Arguments

config	ChartConfig: The chart configuration.
path	Character: Destination file.
complete	Logical: If TRUE, write the output-config form.

Details

`complete = FALSE` (default) writes an **input config**: only the properties that are set. It validates against the chart's `schema.json`.

`complete = TRUE` writes an **output config**: every property, unset ones as explicit nulls, with this package stamped as the `writer`. It validates against `record.json`, and is what an interface hands to another interface – nothing is left for the reader to infer. Resolve the config first, so that the values the data determines are written as the derived facts they are:

```
write_chart_config(resolve(config, data), path, complete = TRUE)
```

Value

`path`, invisibly.

Author(s)

EDG

Examples

```
path <- tempfile(fileext = ".json")
write_chart_config(setup_ScatterConfig(x = "wt", y = "mpg"), path)

# An output config: resolved against the data, then written in full.
cfg <- resolve(setup_ScatterConfig(x = "wt", y = "mpg"), data = mtcars)
write_chart_config(cfg, path, complete = TRUE)
unlink(path)
```

Index

A3Config, 8, *44*, *188*
AnnotationLayer, 10
AreaStyle, 11, *147*, *240*
Axis, 12, *123*
AxisLabel, *13*, 14
AxisLine, *13*, 15
AxisTick, *13*, 16

BarConfig, 17, *44*, *115*, *145*, *189*
BarSeries, 19, *123*
base::jitter(), *71*
BoxplotConfig, 20, *44*, *92*, *115*, *192*
BoxplotSeries, 23, *123*

CalibrationConfig, 25, *44*, *194*
chart_config_to_list, 28
chart_dispatcher_schema, 29
chart_registry, 30
chart_schema, 31
chart_schema(), *7*
ChartConfig, 27, *28, 29*, *31*, *33, 34*, *42–44*, *166*, *168*, *263*
ChoroplethConfig, 32, *44*, *196*
compile, 33
compile(), *44*, *168*
confusion_input, 36
confusion_input(), *60*
ConfusionConfig, 34, *44*, *198*

DataZoom, 37, *52*, *55*, *63*, *78*, *83*, *106*, *123*
DensityConfig, 40, *44*, *200*
draw, 42
draw(), *7, 8*, *17*, *21*, *25*, *27*, *32*, *34, 35*, *40*, *59*, *69*, *86, 87*, *95*, *110*, *125*, *131*, *135*, *144*, *148, 149*, *154*, *157*, *160*, *162*, *168, 169*, *173*, *179*, *181*, *183, 184*, *187*, *189*, *191*, *194*, *196*, *198, 199*, *201*, *204*, *206*, *209*, *211*, *213*, *215*, *217*, *219, 220*, *223*, *226*, *228*, *231–233*, *236*, *239*, *245*, *255*
draw_a3, 45
draw_a3(), *6*, *98, 99*
draw_add_fit, 47
draw_add_fit(), *6*
draw_add_surface, 48
draw_add_surface(), *6*
draw_annotate, 49
draw_annotate(), *6*, *121*
draw_bar, 50
draw_bar(), *6*
draw_boxplot, 52
draw_boxplot(), *6*, *22, 23*, *93*, *117, 118*, *192*
draw_calibration, 56
draw_calibration(), *6*
draw_choropleth, 57
draw_choropleth(), *6*, *86*, *149, 150*
draw_confusion, 60
draw_confusion(), *6*
draw_density, 61
draw_density(), *6*, *78*
draw_fit, 63
draw_fit(), *6*, *47*
draw_gantt, 66
draw_gantt(), *6*, *123*
draw_graph, 68
draw_graph(), *234*
draw_harmonograph, 70
draw_heatmap, 72
draw_heatmap(), *6*, *43*, *133*
draw_histogram, 76
draw_histogram(), *6*
draw_learning_curve, 79
draw_learning_curve(), *6*
draw_line, 81
draw_line(), *6*, *51*, *62*, *74*, *77*, *106*
draw_manhattan, 84
draw_manhattan(), *6*, *100*
draw_map, 86
draw_map(), *149*

draw_mermaid, 88
draw_mermaid(), *6*, *155*, *161*
draw_metric, 91
draw_metric(), *6*
draw_network, 93
draw_network(), *6*, *68*, *128*, *234*
draw_panels, 96
draw_panels(), *6*, *48*, *179*
draw_pie, 97
draw_pie(), *6*
draw_protein, 98
draw_protein(), *6*
draw_pvals, 99
draw_pvals(), *6*
draw_roc, 100
draw_roc(), *6*, *56*
draw_sankey, 102
draw_sankey(), *6*
draw_scatter, 104
draw_scatter(), *6*, *47*, *65*, *82*, *181*, *183, 184*, *220–222*, *224*
draw_scatter3d, 107
draw_scatter3d(), *6*
draw_spectrogram, 108
draw_spectrogram(), *6*
draw_survfit, 111
draw_survfit(), *6*, *112*
draw_survival, 112
draw_survival(), *6*, *111, 112*
draw_ts, 114
draw_ts(), *6*
draw_varimp, 115
draw_varimp(), *6*, *242*
draw_violin, 118
draw_violin(), *6*
draw_volcano, 118
draw_volcano(), *6*
draw_xt, 120
draw_xt(), *6*
drawOutput, 44
drawOutput(), *7*

EChartsOption, *33, 34*, *42, 43*, 122, *148*, *160*, *233*, *260*

FitLayer, 124

GanttConfig, *44*, 125, *202*
GraphEdge, 127, *128*
graphics::hist(), *76*
GraphModel, *68, 69*, 128, *233, 234*
GraphNode, *128*, 128
Grid, *46*, *123*, 129

hclust(), *75*
HeatmapConfig, *44*, 130, *204*
HeatmapSeries, *123*, 133
HistogramConfig, *44*, 134, *207*

ItemStyle, *20*, *24*, *134*, 137, *142*, *147*, *152, 153*, *165*, *175, 176*, *178*, *186*

jsonlite::toJSON(), *29*, *31*, *260*

LabelLine, 139, *166*
LabelOption, *20*, *24*, *134*, 140, *147*, *152, 153*, *163*, *166*, *174–176*, *178*, *186*
Legend, *122*, 141
LineConfig, *44*, *79*, 143, *210*
LineSeries, *123*, 145
LineStyle, *15, 16*, *139*, *147*, 147, *155, 156*, *174, 175*, *178*, *241*

MapLibreOption, *42, 43*, *86*, 148, *160*
MapModel, *86, 87*, *148, 149*, 150
MapRow, *150*, 151
MarkArea, *147*, 152
MarkAreaDataPoint, *152*, 153
MermaidConfig, *44*, 154, *211*
mgcv::gam(), *64*, *105*
MinorSplitLine, *13*, 155
MinorTick, *13*, 156

NetworkConfig, *44*, 157, *213*

panel_layout_schema, 159
PanelLayout, *96*, 159
PeleOption, *42, 43*, 160
PieConfig, *44*, 161, *215*
PieLabelOption, 163
PieSeries, *123*, 164

read_chart_config, 166
read_chart_config(), *7*, *30*
renderDraw, 167
renderDraw(), *7*
resolve, 168
resolve(), *34*
roc_probabilities, 171

roc_vertices, 172
ROCConfig, *44*, 169, *217*
rtemis.a3::create_A3(), *45*
rtemis.a3::read_A3json(), *45*
rtemis.draw *(*rtemis.draw-package*)*, 6
rtemis.draw-package, 6

SankeyConfig, *44*, 172, *219*
SankeyEdgeItem, 174, *176*, *178*
SankeyLevelOption, 175, *178*
SankeyNodeItem, 176, *177*
SankeySeries, 177
save_drawing, 179
save_drawing(), *7*, *43*, *45, 46*, *51*, *54*, *62*, *67*, *71*, *75*, *77*, *80*, *83*, *89*, *92*, *96*, *98*, *103*, *106*, *108*, *111*, *114*, *116*, *121*
Scatter3DConfig, *44*, 180
ScatterConfig, *18*, *44*, *65*, 182, *224*
ScatterSeries, *123*, 185
setup_A3Config, 186
setup_A3Config(), *8*
setup_BarConfig, 188
setup_BarConfig(), *17*, *116*
setup_BoxplotConfig, 190
setup_BoxplotConfig(), *20*, *92, 93*, *116, 117*
setup_CalibrationConfig, 193
setup_CalibrationConfig(), *56*
setup_ChoroplethConfig, 195
setup_ChoroplethConfig(), *32*
setup_ConfusionConfig, 196
setup_ConfusionConfig(), *60*
setup_DensityConfig, 198
setup_DensityConfig(), *40*
setup_GanttConfig, 200
setup_GanttConfig(), *125*
setup_HeatmapConfig, 202
setup_HeatmapConfig(), *130*
setup_HistogramConfig, 205
setup_HistogramConfig(), *134*
setup_LineConfig, 208
setup_LineConfig(), *80*, *143*
setup_MermaidConfig, 210
setup_MermaidConfig(), *6*, *154*
setup_NetworkConfig, 212
setup_NetworkConfig(), *157*
setup_PanelLayout, 214
setup_PieConfig, 214
setup_PieConfig(), *161*
setup_ROCConfig, 216
setup_ROCConfig(), *101*
setup_SankeyConfig, 218
setup_SankeyConfig(), *172*
setup_Scatter3DConfig, 219
setup_Scatter3DConfig(), *108*
setup_ScatterConfig, 221
setup_ScatterConfig(), *7*, *182*
setup_SignificanceConfig, 224
setup_SignificanceConfig(), *85, 86*, *119, 120*
setup_SpectrogramConfig, 227
setup_SpectrogramConfig(), *238*
setup_SurvivalConfig, 229
setup_SurvivalConfig(), *113*
setup_TimeSeriesConfig, 231
setup_TimeSeriesConfig(), *114*, *121*
SigmaOption, *42, 43*, *148*, *160*, 233
signal::specgram(), *108–110*
SignificanceConfig, *44*, *84*, *99*, *118*, *227*, 235
SpectrogramConfig, *44*, *229*, 238
SplitArea, *13*, 240
SplitLine, *13*, 241
stats::density(), *61*
stats::dist(), *73*
stats::glm(), *64*, *105*
stats::hclust(), *73*
stats::p.adjust.methods, *85*, *100*, *119*, *225*, *236*
summarize_varimp, 242
summarize_varimp(), *117*
SurfaceLayer, 243
SurvivalConfig, *44*, *231*, 244

TextStyle, *13, 14*, *123*, *140*, *142*, *164*, 246, *250*, *257*, *259*
Theme, *34*, *42, 43*, *46*, *51*, *54*, *57*, *59, 60*, *62*, *67*, *69*, *71*, *74*, *77*, *80*, *83*, *85*, *87*, *89*, *92*, *95*, *98*, *101*, *103*, *106*, *108*, *110*, *113, 114*, *116*, *120, 121*, 249, *252*, *254*
theme_dark, 250
theme_dark(), *7*
theme_light, 252
theme_light(), *7*, *249*
TimeSeriesConfig, *44*, 254
Title, *122*, 256
to_json, 260

to_list, 260
to_list(), *149*, *159*, *234*
Tooltip, *123*, 258
true_pred_data, 261

varimp_data, 261
VisualMap, *75*, *123*, *133*, 262

write_chart_config, 263
write_chart_config(), *7*, *28*