Package: encharter (via r-universe)

November 1, 2026

Type Package

Title Enhanced Office Open XML Charting for 'openxlsx2'

Version 0.12.1

Language en-US

Description Provides a high-level 'R6' interface for creating complex Office Open XML (OOXML) charts. Allows users to build multi-series combo charts with secondary axes and granular styling options, designed to integrate seamlessly with 'openxlsx2'.

License MIT + file LICENSE

URL https://janmarvin.github.io/encharter/, https://github.com/JanMarvin/encharter

BugReports https://github.com/JanMarvin/encharter/issues

Encoding UTF-8

LinkingTo openxlsx2

Imports R6, grDevices, grid, openxlsx2 (>= 1.26), stats, utils

Suggests testthat (>= 3.0.0), vdiffr, viridisLite, withr

Config/testthat/edition 3

Config/roxygen2/version 8.1.1

Roxygen list(markdown = TRUE)

Config/pak/sysreqs libicu-dev

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

Date/Publication 2026-11-01 10:29:12 UTC

RemoteUrl https://github.com/JanMarvin/encharter

RemoteRef HEAD

RemoteSha 3ac18ba751312f4b93558752fda1438a6b31da23

Contents

encharter 2
encharter_crtx 20
encharter_load 21
EncharterBase 22
plot.Chart 32

Index **34**

`encharter` *Create an Encharter Chart*

Description

Factory function that initialises an R6 chart object. Returns a `Chart` object for standard OOXML chart types (bar, line, scatter, ...) or a `ChartEx` object for modern extended chart types (waterfall, treemap, ...).

The `Chart` class provides a flexible interface to build Office OpenXML (OOXML) chart objects. It allows for granular control over grid lines, secondary axes, and combined chart types (e.g., Bar and Line) within a single plot area.

An R6 class to create and manipulate Office OpenXML (OOXML) Extended Charts (ChartEx), including Waterfall, Sunburst, Treemap, and Region Maps, which are not supported by standard Office Open XML chart types.

Usage

```
encharter(type = "lineChart")

ec(type = "lineChart")
```

Arguments

`type`	A character string specifying the chart type. Common R-style aliases are accepted (see Details).

Details

Supported Chart Types:

- **Bar/Column:** `"barChart"`, `"barplot"`, `"hist"`, `"histogram"`
- **Line/Area:** `"lineChart"`, `"line"`, `"areaChart"`, `"area"`
- **Scatter:** `"scatterChart"`, `"scatter"`, `"point"`
- **Pie/Doughnut:** `"pieChart"`, `"pie"`, `"doughnutChart"`, `"doughnut"`
- **Pie of Pie / Bar of Pie:** `"ofPieChart"`, `"pieOfPie"`, `"barOfPie"` (the latter preselects the bar subtype; see `$set_of_pie_options()`)

- **3D:** "bar3DChart", "line3DChart", "pie3DChart", "area3DChart", "surface3DChart" (see $set_3d_options())
- **Extended (ChartEx):** "waterfall", "treemap", "sunburst", "regionMap", "boxWhisker" / "boxplot", "funnel"

Bar vs Column direction: For bar/column charts, orientation is set via the dir argument in $add_series(): "col" (vertical, default) or "bar" (horizontal).

This class is designed to work with the openxlsx2 package by generating the underlying XML required for the add_chart_xml method.

This class uses XML to manipulate the underlying XML structure and integrates with openxlsx2 for workbook generation.

Value

An R6 object of class Chart or ChartEx.

Further examples

Additional runnable example scripts ship in inst/examples. Each file defines a single function (named after the file) that builds a workbook and opens it in interactive sessions. List or run them with:

```
list.files(system.file("examples", package = "encharter"))
source(system.file("examples", "Bar_Line_Chart.R", package = "encharter"))
```

The available files are:

- 01_Chart_examples.R — tour of standard + extended types in one wb
- All_chartex.R — every ChartEx type (waterfall, sunburst, treemap, ...)
- Axis_labels.R — negative-value bar with axis crossing logic
- BW_with_args.R — box-whisker visibility toggles
- Bar_Area_Chart.R — bars + area combo
- Bar_Line_Chart.R — bars + dashed line on secondary axis
- Bar_Line_and_Data_Table.R — date axis + data table below chart
- Bar_Line_and_Line.R — two independent line/bar demos
- Bar_chart2.R — area-base combo with chart/plot styling
- Bubble_Doughnut.R — doughnut + bubble on one sheet
- Chart_and_plot_style.R — chart-area vs plot-area styling
- Droplines_highlowlines_updownbars.R — line adornments
- Histogram_with_args.R — histogram via clusteredColumn binning
- Label_Grouping.R — multi-level category labels
- Load_and_extend.R — load a chart back from a workbook and extend its range
- Chart_template.R — save a chart as a .crtx template, read and apply it
- Line.R — line with markers and global data labels

- New_chart_types.R — 0.11 showcase: pie/bar of pie, all 3D types, display units, tick skips, directional error bars
- Pie.R — pie with viridis palette
- Radar_chart.R — standard vs filled radar
- Scatter.R — markers-only scatter
- Seatbelts.R — Seatbelts time series with rolling rates
- StockCharts.R — stockChart with high/low and up/down bars
- Styled_Bars.R — heavy series + axis + grid styling
- Surface_Plot.R — surface (contour) plot from a matrix
- Treemap_with_args.R — treemap with parent_label = "banner"
- Trendline_and_errorbars.R — series error bars + linear trendline
- Waterfall.R — financial bridge with subtotal
- Waterfall2.R — waterfall with date X-axis
- Waterfall3.R — fully themed waterfall
- line_scatterplot.R — multi-species scatter from iris

Run all of them in one session with `run_all_examples()` (defined in `inst/examples/run_all_examples.R`).

Super class

`EncharterBase` -> `Chart`

Public fields

`x2_title` List containing text and style for the secondary X-axis.

`y2_title` List containing text and style for the secondary Y-axis.

`first_slice_ang` Integer. Rotation of the first slice (0-360).

`expansion` Integer. Size of the expansion for pie charts.

`hole_size` Integer. Size of the hole for doughnut charts (0-90).

`show_data_table` Logical if a data table should be added.

`drop_lines` Logical; show lines from points to the axis.

`high_low_lines` Logical; show lines between max/min points.

`up_down_bars` Logical; show bars between first and last series.

`bubble_scale` Numeric; the scale factor for bubbles (default 100).

`show_neg_bubbles` Logical; whether to show bubbles with negative values.

`disp_blanks_as` Character; "gap", "span", or "zero".

`plot_vis_only` Logical; plot only cells in visible rows and columns (default `TRUE`).

`of_pie_type` Character; subtype of `ofPieChart`: "pie" or "bar".

`second_pie_size` Integer; size of the second pie/bar plot as a percentage (5-200) for `ofPieChart`.

`split_type` Character; how points are split into the second plot for `ofPieChart`: "auto", "cust", "percent", "pos", or "val".

`split_pos` Numeric; split threshold, or point indices (0-based) when `split_type = "cust"`.

`view3d` Named list of 3D view parameters (`rot_x`, `rot_y`, `perspective`, `depth_percent`, `h_percent`, `right_angle_axes`).

`gap_depth` Integer; gap depth percentage (0-500) for 3D charts.

`bar_shape` Character; bar shape for `bar3DChart`: "box", "cylinder", "cone", "coneToMax", "pyramid", or "pyramidToMax".

`size_represents` Character; bubble size meaning, "area" or "w".

`template` List of series styles from a chart template; see `encharter_from_crtx()`.

`plot_layout` Fixed position of the plot area as fractions of the chart (`x`, `y`, `w`, `h`, `target`), or `NULL` for automatic layout.

`text_style` Default text properties of the chart (`font_size`, `font_name`, `font_color`, `bold`, `italic`), used by text without its own.

Methods

Public methods:

- `Chart$new()`
- `Chart$set_x2_title()`
- `Chart$set_y2_title()`
- `Chart$set_y2_axis()`
- `Chart$set_x2_axis()`
- `Chart$set_data_table()`
- `Chart$set_pie_options()`
- `Chart$set_of_pie_options()`
- `Chart$set_3d_options()`
- `Chart$set_bubble_options()`
- `Chart$set_disp_blanks()`
- `Chart$set_plot_vis_only()`
- `Chart$add_series()`
- `Chart$apply_crtx()`
- `Chart$update_series()`
- `Chart$render()`
- `Chart$clone()`

`Chart$new()`: Initialize a new Chart object.

Usage:

```
Chart$new(type = NULL)
```

Arguments:

`type` Initial chart type (e.g., "lineChart", "barChart", "pieChart").

`Chart$set_x2_title()`: Set the secondary X-axis title.

Only takes effect if at least one series has been assigned to the secondary X-axis via `add_series(secondary = "x")`. Issues a warning and returns `self` silently otherwise.

Usage:

```
Chart$set_x2_title(
  text,
  font_size = NULL,
  font_name = NULL,
  font_color = NULL,
  bold = NULL,
  italic = NULL,
  fill = NULL,
  line = NULL,
  line_width = NULL
)
```

Arguments:

`text` Title string.

`font_size` Numeric font size in points.

`font_name` Font typeface name.

`font_color` Six-digit hex color for the title text.

`bold`, `italic` Logical font style.

`fill` Six-digit hex color for the title background box.

`line` Six-digit hex color for the title border.

`line_width` Numeric border width in points.

Examples:

```
ec("scatter")$
  add_series(data = "Sheet1!A1:A10", secondary = "x")$
  set_x2_title("Secondary X", font_color = "888888")
```

`Chart$set_y2_title()`**:** Set the secondary Y-axis title.

Only takes effect if at least one series has been assigned to the secondary Y-axis via `add_series(secondary = TRUE)` or `secondary = "y"`. Issues a warning otherwise.

Usage:

```
Chart$set_y2_title(
  text,
  font_size = NULL,
  font_name = NULL,
  font_color = NULL,
  bold = NULL,
  italic = NULL,
  fill = NULL,
  line = NULL,
  line_width = NULL
)
```

Arguments:

`text` Title string.

`font_size` Numeric font size in points.

`font_name` Font typeface name.

`font_color` Six-digit hex color for the title text.

`bold, italic` Logical font style.

`fill` Six-digit hex color for the title background box.

`line` Six-digit hex color for the title border.

`line_width` Numeric border width in points.

Examples:

```
ec("line")$
  add_series(data = "Sheet1!A1:A10")$
  add_series(data = "Sheet1!B1:B10", secondary = TRUE)$
  set_y2_title("Growth Rate (%)")
```

`Chart$set_y2_axis()`: Set Secondary Y-axis scaling, units, and format. Arguments that are not given keep their current value, so the method can be called several times.

Usage:

```
Chart$set_y2_axis(
  min = NULL,
  max = NULL,
  major = NULL,
  minor = NULL,
  major_time = NULL,
  minor_time = NULL,
  base_time = NULL,
  major_tick = NULL,
  minor_tick = NULL,
  format = NULL,
  log_base = NULL,
  rev = NULL,
  color = NULL,
  font_name = NULL,
  font_size = NULL,
  bold = NULL,
  italic = NULL,
  font_color = NULL,
  rotation = NULL,
  grid_color = NULL,
  grid_lines = NULL,
  minor_grid_color = NULL,
  minor_grid_lines = NULL,
  cross_between = NULL,
  line_width = NULL,
  grid_width = NULL,
  minor_grid_width = NULL,
  crosses = NULL,
  crosses_at = NULL,
  label_pos = NULL,
  tick_lbl_skip = NULL,
  tick_mark_skip = NULL,
```

```
  disp_units = NULL
)
```

Arguments:

`min` Minimum value for the axis.

`max` Maximum value for the axis.

`major` Numeric value for major unit interval.

`minor` Numeric value for minor unit interval.

`major_time` Time unit for major steps ("days", "months", "years"). Used for date axes.

`minor_time` Time unit for minor steps ("days", "months", "years"). Used for date axes.

`base_time` Base time unit for date axes ("days", "months", "years").

`major_tick, minor_tick` Tick marks for major and minor ("cross", "in", "none", "out").

`format` A number format string (e.g., "#,##0" or "yyyy-mm-dd").

`log_base` Base for logarithmic scaling (e.g., 10).

`rev` Logical to reverse the value order

`color, font_color` Hex color for the axis lines and label (or independent label color).

`font_name` Font typeface name (e.g., "Arial", "Calibri").

`font_size` Font size for the axis labels.

`bold` Logical; if `TRUE`, axis labels will be bold.

`italic` Logical; if `TRUE`, axis labels will be italicized.

`rotation` Rotation in degrees.

`grid_color, minor_grid_color` Hex color for the grid lines.

`grid_lines, minor_grid_lines` Logical. Show or hide grid lines.

`cross_between` Specifies how the value axis crosses the category axis ('between' or 'midCat').

`line_width, grid_width, minor_grid_width` Numeric. Change the width of the axis and grid lines.

`crosses` Intersection: "autoZero", "min" (start), or "max" (end, default).

`crosses_at` Numeric axis value for intersection. Overrides 'crosses'.

`label_pos` Label position: "nextTo" (default), "low" (edge of chart), "high" (opposite edge), or "none".

`tick_lbl_skip, tick_mark_skip` Integer (>= 1); label/tick every n-th category (category axes only).

`disp_units` Display units: a built-in unit string (e.g. "thousands") or a positive number (value axes only).

`Chart$set_x2_axis()`: Set Secondary X-axis scaling, units, and format. Arguments that are not given keep their current value, so the method can be called several times.

Usage:

```
Chart$set_x2_axis(
  min = NULL,
  max = NULL,
  major = NULL,
  minor = NULL,
  major_time = NULL,
  minor_time = NULL,
```

```
  base_time = NULL,
  major_tick = NULL,
  minor_tick = NULL,
  format = NULL,
  log_base = NULL,
  rev = NULL,
  color = NULL,
  font_name = NULL,
  font_size = NULL,
  bold = NULL,
  italic = NULL,
  font_color = NULL,
  rotation = NULL,
  grid_color = NULL,
  grid_lines = NULL,
  minor_grid_color = NULL,
  minor_grid_lines = NULL,
  cross_between = NULL,
  line_width = NULL,
  grid_width = NULL,
  minor_grid_width = NULL,
  crosses = NULL,
  crosses_at = NULL,
  label_pos = NULL,
  tick_lbl_skip = NULL,
  tick_mark_skip = NULL,
  disp_units = NULL
)
```

Arguments:

`min` Minimum value for the axis.

`max` Maximum value for the axis.

`major` Numeric value for major unit interval.

`minor` Numeric value for minor unit interval.

`major_time` Time unit for major steps ("days", "months", "years"). Used for date axes.

`minor_time` Time unit for minor steps ("days", "months", "years"). Used for date axes.

`base_time` Base time unit for date axes ("days", "months", "years").

`major_tick, minor_tick` Tick marks for major and minor ("cross", "in", "none", "out").

`format` A number format string (e.g., "#,##0" or "yyyy-mm-dd").

`log_base` Base for logarithmic scaling (e.g., 10).

`rev` Logical to reverse the value order

`color, font_color` Hex color for the axis lines and label (or independent label color).

`font_name` Font typeface name (e.g., "Arial", "Calibri").

`font_size` Font size for the axis labels.

`bold` Logical; if `TRUE`, axis labels will be bold.

`italic` Logical; if `TRUE`, axis labels will be italicized.

`rotation` Rotation in degrees.

`grid_color, minor_grid_color` Hex color for the grid lines.

`grid_lines, minor_grid_lines` Logical. Show or hide grid lines.

`cross_between` Specifies how the value axis crosses the category axis ('between' or 'midCat').

`line_width, grid_width, minor_grid_width` Numeric. Change the width of the axis and grid lines.

`crosses` Intersection: "autoZero", "min" (start), or "max" (end, default).

`crosses_at` Numeric axis value for intersection. Overrides 'crosses'.

`label_pos` Label position: "nextTo" (default), "low" (edge of chart), "high" (opposite edge), or "none".

`tick_lbl_skip, tick_mark_skip` Integer (>= 1); label/tick every n-th category (category axes only).

`disp_units` Display units: a built-in unit string (e.g. "thousands") or a positive number (value axes only).

`Chart$set_data_table()`: Set the data table.

Usage:

```
Chart$set_data_table(show = TRUE)
```

Arguments:

`show` Logical TRUE or FALSE.

`Chart$set_pie_options()`:

Usage:

```
Chart$set_pie_options(rotation = NULL, expansion = NULL, hole_size = NULL)
```

Arguments:

`rotation` The angle of the first slice in degrees, from 0 to 360. This rotates the chart clockwise.

`expansion` Sets the expansion, from 0 to 400.

`hole_size` Set the hole size of (only doughnut charts), from 0 to 90.

`Chart$set_of_pie_options()`: Configure the Pie of Pie / Bar of Pie chart (`ofPieChart`).

Usage:

```
Chart$set_of_pie_options(
  type = NULL,
  second_size = NULL,
  split_type = NULL,
  split_pos = NULL
)
```

Arguments:

`type` Subtype: `"pie"` (Pie of Pie, default) or `"bar"` (Bar of Pie).

`second_size` Size of the second plot as a percentage of the main pie, from 5 to 200. Default 75.

`split_type` How data points are assigned to the second plot: `"auto"` (default), `"percent"`, `"pos"` (last n points), `"val"` (values below threshold), or `"cust"`.

`split_pos` Numeric split threshold for `"percent"`, `"pos"`, and `"val"`; for `"cust"` a vector of 0-based point indices to move to the second plot.

Examples:

```
ec("ofPieChart")$set_of_pie_options(type = "bar", split_type = "pos", split_pos = 3)
```

`Chart$set_3d_options()`: Configure the 3D view and 3D-only chart options. Only takes effect for the 3D chart types (`bar3DChart`, `line3DChart`, `pie3DChart`, `area3DChart`, `surface3DChart`) and `surfaceChart`.

Usage:

```
Chart$set_3d_options(
  rot_x = NULL,
  rot_y = NULL,
  perspective = NULL,
  depth_percent = NULL,
  h_percent = NULL,
  right_angle_axes = NULL,
  gap_depth = NULL,
  shape = NULL
)
```

Arguments:

`rot_x` Rotation around the X-axis in degrees, from -90 to 90.

`rot_y` Rotation around the Y-axis in degrees, from 0 to 360.

`perspective` Perspective in half-degrees, from 0 to 240 (ignored when `right_angle_axes = TRUE`).

`depth_percent` Depth as a percentage of chart width, 20 to 2000.

`h_percent` Height as a percentage of chart width, 5 to 500.

`right_angle_axes` Logical; render axes at right angles instead of in perspective.

`gap_depth` Gap depth percentage between series, 0 to 500 (bar/line/area 3D).

`shape` Bar shape for `bar3DChart`: `"box"` (default), `"cylinder"`, `"cone"`, `"coneToMax"`, `"pyramid"`, or `"pyramidToMax"`.

Examples:

```
ec("bar3DChart")$set_3d_options(rot_x = 20, rot_y = 30, shape = "cylinder")
```

`Chart$set_bubble_options()`:

Usage:

```
Chart$set_bubble_options(scale = 100, show_neg = FALSE, size_represents = NULL)
```

Arguments:

`scale` The scale factor for bubbles, from 0 to 300 (expressed as a percentage).

`show_neg` Logical; if `TRUE`, bubbles with negative values will be displayed on the chart.

`size_represents` What the bubble size encodes: `"area"` (default in Excel) or `"w"` (width/diameter). `NULL` omits the element.

`Chart$set_disp_blanks()`: Set missing value behavior ("gap", "span", "zero").

Usage:

```
Chart$set_disp_blanks(val = "gap")
```

Arguments:

`val` Character. One of "gap" (break), "span" (continue), or "zero" (drop).

`Chart$set_plot_vis_only()`: Set whether data in hidden rows and columns is plotted.

Usage:

```
Chart$set_plot_vis_only(val = TRUE)
```

Arguments:

`val` Logical. If `TRUE` (default), only visible cells are plotted.

`Chart$add_series()`: Add a data series to the chart with independent styling.

Usage:

```
Chart$add_series(
  name = NULL,
  data,
  label = NULL,
  weight = NULL,
  color = "4472C4",
  type = NULL,
  secondary = FALSE,
  dir = "col",
  grouping = "standard",
  overlap = NULL,
  gap_width = NULL,
  smooth = FALSE,
  show_line = TRUE,
  marker = "none",
  marker_size = 5,
  marker_fill = NULL,
  marker_line = NULL,
  marker_line_width = 0.75,
  show_val = NULL,
  show_cat = NULL,
  line_type = NULL,
  line_width = 1,
  line_color = NULL,
  filled = FALSE,
  error_bars = FALSE,
  trendline = FALSE,
  invert_if_negative = FALSE
)
```

Arguments:

`name` Cell range or string for series name.

`data` Cell range for series values.

`label` Cell range for category labels.

`weight` Cell range for bubble sizes (bubbleChart only).

`color` Primary Hex color for the series (used as default for line and markers). `NULL` takes the next color of the chart's `palette`.

`type` Chart type for this specific series (for combo charts).

`secondary` Logical. Set to TRUE to move series to secondary axis.

`dir` Bar direction ("col" or "bar").

`grouping` Chart grouping ("standard", "stacked", "percentStacked").

`overlap` Integer between -100 and 100 for bar charts.

`gap_width` Integer between 0 and 500 for bar charts.

`smooth` Logical. Enable line smoothing for line/scatter charts.

`show_line` Logical. Show the line connecting points.

`marker` Marker type ("none", "circle", "square", "diamond", "triangle").

`marker_size` Integer size of marker.

`marker_fill` Hex color for the interior of the marker. Defaults to `color`.

`marker_line` Hex color for the marker border. Defaults to `color`.

`marker_line_width` Numeric width of the marker border.

`show_val` Logical. Override global label settings for this series (show value).

`show_cat` Logical. Override global label settings for this series (show category).

`line_type` Line style: "dashed", "dotted", "dashDot", or "solid".

`line_width` Numeric width of the connecting line.

`line_color` Hex color for the connecting line. Defaults to `color`.

`filled` Logical; for radar charts, fills the interior area. Default FALSE.

`error_bars` A list of error bar properties:

- `type`: The error value type (`ST_ErrValType`). Must be one of: `"fixedVal"` (Fixed Value), `"percentage"` (Percentage), `"stdDev"` (Standard Deviation), `"stdErr"` (Standard Error), or `"cust"` (Custom).
- `value`: The numeric value for the error bars (e.g., 10 for 10% or 5 for fixed units).
- `direction`: Direction of bars. One of `"both"`, `"plus"`, or `"minus"`.
- `axis`: Error direction axis, `"y"` (default) or `"x"` (horizontal bars, scatter charts).
- `color`: Hex color code for the bars (e.g., "FF0000").

`trendline` A list of regression line properties:

- `type`: The regression type (`ST_TrendlineType`). Must be one of: `"linear"` (Linear), `"exp"` (Exponential), `"log"` (Logarithmic), `"movingAvg"` (Moving Average), `"poly"` (Polynomial), or `"power"` (Power).
- `order`: Required for `"poly"`; an integer between 2 and 6.
- `period`: Required for `"movingAvg"`; an integer representing the window size.
- `forward`, `backward`: Numeric; extrapolate the line n periods forwards/backwards.
- `intercept`: Numeric; force the line through a fixed y-intercept.
- `color`: Hex color code for the line.
- `show_r2`: Logical; if `TRUE`, displays the R-squared value on the chart.

`invert_if_negative` Logical; bar charts only. Invert the fill for negative values. Default `FALSE`.

`Chart$apply_crtx()`**:** Apply the styling of a chart template (`.crtx`) to this chart: chart and plot area, title and legend style, axes, and the styling of the series in order. Series data, ranges and titles are kept. Series added afterwards take the template styling as well.

Usage:

```
Chart$apply_crtx(path)
```

Arguments:

`path` Path of the `.crtx` file.

Returns: The chart, invisibly.

`Chart$update_series()`: Change an existing series. Takes the arguments of `add_series()`; arguments that are not supplied keep their current value. With a `wb_data()` object as `data` and no `name`, the column is found from the series' current header (or data) cell, so `update_series(data = wb_data(wb), label = Month)` re-points every series at the current extent of the data.

Usage:

```
Chart$update_series(
  index = NULL,
  name = NULL,
  data = NULL,
  label = NULL,
  weight = NULL,
  color = NULL,
  type = NULL,
  secondary = NULL,
  dir = NULL,
  grouping = NULL,
  overlap = NULL,
  gap_width = NULL,
  smooth = NULL,
  show_line = NULL,
  marker = NULL,
  marker_size = NULL,
  marker_fill = NULL,
  marker_line = NULL,
  marker_line_width = NULL,
  line_type = NULL,
  line_width = NULL,
  line_color = NULL,
  filled = NULL,
  error_bars = NULL,
  trendline = NULL,
  invert_if_negative = NULL
)
```

Arguments:

`index` Integer vector of series to update. Default: all series.

`name`, `data`, `label`, `weight`, `color`, `type`, `secondary`, `dir`, `grouping`, `overlap`, `gap_width`, `smooth`, `show_lin`
See `add_series()`.

Examples:

```
wb <- openxlsx2::wb_workbook()$add_worksheet("Data")$add_data(
  x = data.frame(Month = month.abb[1:6], Sales = 1:6))
```

```
chart <- ec("line")$add_series(name = Sales, data = openxlsx2::wb_data(wb), label = Month)
wb$add_data(x = data.frame(Month = "Jul", Sales = 7), dims = "A8", col_names = FALSE)
chart$update_series(data = openxlsx2::wb_data(wb), label = Month, color = "C00000")
```

`Chart$render()`: Generate the final XML string for the chart.

Usage:

```
Chart$render(
  u_ids = c("53178645", "60812428", "64752656", "81893617", "90007639")
)
```

Arguments:

`u_ids` five unique ids

Returns: A character string containing the OOXML chart definition.

`Chart$clone()`: The objects of this class are cloneable with this method.

Usage:

```
Chart$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Super class

`EncharterBase` -> `ChartEx`

Public fields

`color_xml` color

`style_xml` style

`region_colors` Internal; color scale entries for region maps set via `set_region_map_colors()`.

Methods

Public methods:

- `ChartEx$new()`
- `ChartEx$set_waterfall_colors()`
- `ChartEx$set_color_cycle()`
- `ChartEx$set_region_map_colors()`
- `ChartEx$add_series()`
- `ChartEx$update_series()`
- `ChartEx$render()`
- `ChartEx$clone()`

`ChartEx$new()`: Create a new ChartEx object.

Usage:

```
ChartEx$new(type = NULL)
```

Arguments:

`type` Initial chart type (e.g., "waterfall", "treemap").

Returns: A new `ChartEx` object.

`ChartEx$set_waterfall_colors()`: Set the semantic waterfall colors (also used as the first colors of the chart's color cycle). ChartEx charts derive both the point fills and the legend keys from the chart's color style part (`colors{n}.xml`); waterfall maps Increase, Decrease, and Total to the first three entries of that cycle. This method replaces those entries, so the bars and the legend stay consistent. For individual outlier points (e.g. an "unexpected decrease"), pass a per-point `color` vector to `add_series()` instead.

Usage:

```
ChartEx$set_waterfall_colors(increase = NULL, decrease = NULL, total = NULL)
```

Arguments:

`increase` Fill for rising values: hex, an R color name, or `openxlsx2::wb_color()` (theme colors supported).

`decrease` Fill for falling values.

`total` Fill for subtotal/total points.

Examples:

```
ec("waterfall")$
  add_series(data = "Sheet1!B2:B7", label = "Sheet1!A2:A7", subtotals = c(0, 5))$
  set_waterfall_colors(increase = "70AD47", decrease = "C00000", total = "A6A6A6")
```

`ChartEx$set_color_cycle()`: Set the chart's color cycle (the color style part, `colors{n}.xml`). ChartEx charts derive series, category, and legend colors from this cycle: treemap and sunburst color their top-level categories from it, box & whisker and histogram color their series, funnel and Pareto take their first colors from it. The first `length(colors)` cycle entries are replaced in order; if more colors are given than the part contains, the cycle is extended. Remaining entries and the brightness variations are kept.

Usage:

```
ChartEx$set_color_cycle(colors)
```

Arguments:

`colors` Character vector of colors (hex or R color names), or a list which may also contain `openxlsx2::wb_color()` values (theme colors supported).

Examples:

```
ec("treemap")$
  add_series(data = "Sheet1!B2:B7", label = "Sheet1!A2:A7")$
  set_color_cycle(c("C00000", "4472C4", "70AD47", "FFC000"))
```

`ChartEx$set_region_map_colors()`: Set the color scale of a region map. Written as the series' `cx:valueColors` element, which drives both the map shading and the legend's color scale. Only applied to `regionMap` series.

Usage:

```
ChartEx$set_region_map_colors(min, max, mid = NULL)
```

Arguments:

`min` Color for the smallest values (hex, R color name, or `openxlsx2::wb_color()`).

`max` Color for the largest values.

`mid` Optional middle color for a three-color scale.

Examples:

```
ec("regionMap")$
  add_series(data = "Sheet1!B2:B7", label = "Sheet1!A2:A7")$
  set_region_map_colors(min = "FFF2CC", max = "C00000")
```

`ChartEx$add_series()`**:** Add a data series to the chart.

Usage:

```
ChartEx$add_series(
  name = NULL,
  data,
  label = NULL,
  type = NULL,
  color = "auto",
  line_color = NULL,
  line_width = 1,
  gap_width = NULL,
  subtotals = NULL,
  statistics = NULL,
  binning = NULL,
  visibility = NULL,
  parent_label = "overlapping"
)
```

Arguments:

`name` Cell range for the series name.

`data` Cell range for the numeric values.

`label` Cell range for the category labels.

`type` Type of chart (waterfall, sunburst, treemap, regionMap).

`color` Hex color or "auto". `NULL` takes the next color of the chart's `palette`.

`line_color` Border color.

`line_width` Border width.

`gap_width` Integer between 0 and 500.

`subtotals` Numeric vector of indices to treat as subtotals (Waterfall only).

`statistics` Quartile method: "inclusive" or "exclusive".

`binning` A list for Histogram/BoxWhisker: `binSize` (numeric), `binCount` (integer), `intervalClosed` ("left", "right"), `underflow` (numeric or "auto"), `overflow` (numeric or "auto").

`visibility` A named list of logicals for BoxWhisker/Waterfall: `connectorLines`, `meanLine`, `meanMarker`, `nonoutliers`, `outliers`.

`parent_label` Treemap label style: "overlapping", "banner", or "none".

`ChartEx$update_series()`**:** Change an existing series. Takes the arguments of `add_series()`; arguments that are not supplied keep their current value. With a `wb_data()` object as `data` and no `name`, the column is found from the series' current header (or data) cell.

Usage:

```
ChartEx$update_series(
  index = NULL,
  name = NULL,
  data = NULL,
  label = NULL,
  color = NULL,
  line_color = NULL,
  line_width = NULL,
  gap_width = NULL,
  subtotals = NULL,
  statistics = NULL,
  binning = NULL,
  visibility = NULL,
  parent_label = NULL
)
```

Arguments:

`index` Integer vector of series to update. Default: all series.

`name`, `data`, `label`, `color`, `line_color`, `line_width`, `gap_width`, `subtotals`, `statistics`, `binning`, `visibili`
See `add_series()`.

`ChartEx$render()`**:** Render the internal XML for writing to a file.

Usage:

```
ChartEx$render(id_start = 1, guid = "{C59B1284-E301-0D0F-1B20-FD96A66D6E43}")
```

Arguments:

`id_start` Numeric starting ID for XML data references.

`guid` a guid

Returns: A list containing the XML and attribute mappings.

`ChartEx$clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
ChartEx$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

```
# Standard line chart
ec("lineChart")

# Extended waterfall chart
ec("waterfall")

# R-style alias
ec("barplot")
```

```
## ------------------------------------------------
## Method `Chart$set_x2_title()`
## ------------------------------------------------

ec("scatter")$
  add_series(data = "Sheet1!A1:A10", secondary = "x")$
  set_x2_title("Secondary X", font_color = "888888")

## ------------------------------------------------
## Method `Chart$set_y2_title()`
## ------------------------------------------------

ec("line")$
  add_series(data = "Sheet1!A1:A10")$
  add_series(data = "Sheet1!B1:B10", secondary = TRUE)$
  set_y2_title("Growth Rate (%)")

## ------------------------------------------------
## Method `Chart$set_of_pie_options()`
## ------------------------------------------------

ec("ofPieChart")$set_of_pie_options(type = "bar", split_type = "pos", split_pos = 3)

## ------------------------------------------------
## Method `Chart$set_3d_options()`
## ------------------------------------------------

ec("bar3DChart")$set_3d_options(rot_x = 20, rot_y = 30, shape = "cylinder")

## ------------------------------------------------
## Method `Chart$update_series()`
## ------------------------------------------------

wb <- openxlsx2::wb_workbook()$add_worksheet("Data")$add_data(
  x = data.frame(Month = month.abb[1:6], Sales = 1:6))
chart <- ec("line")$add_series(name = Sales, data = openxlsx2::wb_data(wb), label = Month)
wb$add_data(x = data.frame(Month = "Jul", Sales = 7), dims = "A8", col_names = FALSE)
chart$update_series(data = openxlsx2::wb_data(wb), label = Month, color = "C00000")

## ------------------------------------------------
## Method `ChartEx$set_waterfall_colors()`
## ------------------------------------------------

ec("waterfall")$
  add_series(data = "Sheet1!B2:B7", label = "Sheet1!A2:A7", subtotals = c(0, 5))$
  set_waterfall_colors(increase = "70AD47", decrease = "C00000", total = "A6A6A6")

## ------------------------------------------------
## Method `ChartEx$set_color_cycle()`
## ------------------------------------------------

ec("treemap")$
```

```
  add_series(data = "Sheet1!B2:B7", label = "Sheet1!A2:A7")$
  set_color_cycle(c("C00000", "4472C4", "70AD47", "FFC000"))

## ------------------------------------------------
## Method `ChartEx$set_region_map_colors()`
## ------------------------------------------------

ec("regionMap")$
  add_series(data = "Sheet1!B2:B7", label = "Sheet1!A2:A7")$
  set_region_map_colors(min = "FFF2CC", max = "C00000")
```

---

encharter_crtx *Read and write chart templates (.crtx)*

Description

A chart template is a zip package holding a chart part without data references. Excel writes them from "Save as Template"; other applications read them as well. `encharter_to_crtx()` saves a `Chart` as a template, `encharter_from_crtx()` reads a template into a `Chart` without series, and `Chart$apply_crtx()` copies the styling of a template onto an existing chart.

Usage

```
encharter_to_crtx(chart, path)

ec_to_crtx(chart, path)

encharter_from_crtx(path)

ec_from_crtx(path)
```

Arguments

`chart`	A `Chart` object.
`path`	Path of the `.crtx` file.

Details

A template read with `encharter_from_crtx()` keeps the per-series styling (color, line, marker) of the template. Series added to it with `$add_series()` take that styling in order, for the arguments not given explicitly; when the template has fewer series than added, its styles are reused from the start.

Templates cover standard charts only; extended charts (`ChartEx`) are not supported. Templates written by Excel also carry a theme override and a preview image, which are not read; a template written here holds the chart part only.

Value

`encharter_to_crtx()` returns `path` invisibly; `encharter_from_crtx()` returns a `Chart` object.

Examples

```
wb <- openxlsx2::wb_workbook()$add_worksheet("Data")$add_data(x = data.frame(
  Month = month.abb[1:6], Sales = c(120, 135, 128, 160, 175, 190)
))
chart <- ec("bar")$set_chart_title("Sales", bold = TRUE)$
  set_y_axis(grid_lines = TRUE, grid_color = "EEEEEE")$
  add_series(name = Sales, data = openxlsx2::wb_data(wb), label = Month, color = "2E4057")
tmp <- tempfile(fileext = ".crtx")
encharter_to_crtx(chart, tmp)

chart2 <- encharter_from_crtx(tmp)
chart2$add_series(name = "Data!$B$1", data = "Data!$B$2:$B$7", label = "Data!$A$2:$A$7")
chart2$series_data[[1]]$line$color
```

---

encharter_load *Load a chart from a workbook into an encharter object*

Description

Reads a chart stored in a workbook and returns it as a `Chart` or `ChartEx` object, the counterpart of `openxlsx2::wb_add_encharter()`. Series references and cached values, styling, axes, titles and the legend are restored, so the object can be modified like any other encharter object (`$update_series()` to follow a longer data range, `$add_series()`, `$set_chart_title()`, ...) and written back with `wb_add_encharter()`.

Usage

```
encharter_load(wb, chart = 1L, type = NULL)

ec_load(wb, chart = 1L, type = NULL)
```

Arguments

`wb`	A `wbWorkbook` containing at least one chart.
`chart`	Integer; the row of `wb$charts` to load. Default `1`.
`type`	`"chart"` for a standard chart or `"chartEx"` for an extended chart. The default takes whichever the row holds and errors if it holds both.

Details

Charts created with encharter are reproduced exactly. Charts written by other software load as well; properties encharter has no field for (manual layouts, per-point styling outside pie charts, theme color modifiers, extension lists) are not carried over.

A workbook loaded from a file can hold a standard chart and an extended chart (waterfall, treemap, ...) in the same row of `wb$charts`; use `type` to pick one in that case.

Value

A `Chart` or `ChartEx` object.

Examples

```
library(openxlsx2)
wb <- wb_workbook()$add_worksheet("Data")$add_data(x = data.frame(
  Month = month.abb[1:6],
  Sales = c(120, 135, 128, 160, 175, 190)
))
chart <- ec("line")$set_chart_title("Sales")$
  add_series(name = "Data!$B$1", label = "Data!$A$2:$A$7",
             data = "Data!$B$2:$B$7", marker = "circle")
wb$add_encharter(dims = "D2:L18", graph = chart)

# A month later: append a row and point the chart at the longer range
wb$add_data(x = data.frame(Month = "Jul", Sales = 240), dims = "A8", col_names = FALSE)
chart <- ec_load(wb)
chart$update_series(data = wb_data(wb), label = Month)
wb$add_encharter(dims = "D20:L36", graph = chart)
```

EncharterBase *Encharter Base R6 Class*

Description

Abstract base class inherited by `Chart` and `ChartEx`. Holds all shared fields (palette, titles, axis params, legend/label settings) and the shared private helpers (`render_color_core`, `render_color`, `set_axis_params`, `validate_input`).

Users should not instantiate `EncharterBase` directly; use `encharter()` instead.

Public fields

`xml` The raw xml2 object containing the chart space.

`series_data` A list containing all added data series and their styles.

`type` The default chart type for the object (e.g., `"lineChart"`).

`palette` A character vector of six-digit hex colors. Series added with `color = NULL` take the next color of the palette. In pie, doughnut and pie-of-pie charts, and in a chart with a single bubble series at its default color, the palette colors the data points. Defaults to the standard Office theme palette.

`chart_title` Named list with elements `text` (character) and `style` (list of font/fill/line options) for the main chart title.

`x_title` Named list with elements `text` and `style` for the primary X-axis title.

`y_title` Named list with elements `text` and `style` for the primary Y-axis title.

`chart_style` Named list controlling the outer chart area: `fill` (hex), `line` (hex), `line_width` (numeric).

`plot_style` Named list controlling the inner plot area: `fill` (hex), `line` (hex), `line_width` (numeric).

`label_params` Named list of global data label defaults: `show_val`, `show_cat`, `show_legend_key`, `leader_lines` (logicals), `pos`, `sep`, `fill` (character), `style` (list).

`legend_params` Named list of legend defaults: `pos` (character), `overlay` ("0"/"1"), `style` (list).

`axis_params` Named list with one entry per axis (`x`, `x2`, `y`, `y2`). Each entry is a named list of scaling, formatting, and style parameters. Modified via `$set_x_axis()`, etc.

Methods

Public methods:

- `EncharterBase$set_chart_title()`
- `EncharterBase$set_x_title()`
- `EncharterBase$set_y_title()`
- `EncharterBase$set_x_axis()`
- `EncharterBase$set_y_axis()`
- `EncharterBase$set_data_label_style()`
- `EncharterBase$set_legend_style()`
- `EncharterBase$set_chart_style()`
- `EncharterBase$set_plot_style()`
- `EncharterBase$print()`
- `EncharterBase$clone()`

`EncharterBase$set_chart_title()`: Set the chart's main title.

Usage:

```
EncharterBase$set_chart_title(
  text,
  font_size = NULL,
  font_name = NULL,
  font_color = NULL,
  bold = NULL,
  italic = NULL,
  fill = NULL,
  line = NULL,
  line_width = NULL
)
```

Arguments:

`text` Title string. Accepts a plain character or an `openxlsx2::fmt_txt()` object for rich-text formatting.

`font_size` Numeric font size in points (e.g. 14).

`font_name` Font typeface name (e.g. `"Arial"`).

`font_color` Six-digit hex color for the title text (e.g. `"FF0000"` for red).

`bold` Logical; `TRUE` renders the title in bold.

`italic` Logical; `TRUE` renders the title in italics.

`fill` Six-digit hex color for the title background box.

`line` Six-digit hex color for the title border.

`line_width` Numeric border width in points.

Examples:

```
ec("line")$set_chart_title("Monthly Sales", font_size = 14, bold = TRUE)
```

`EncharterBase$set_x_title()`: Set the primary X-axis title.

Usage:

```
EncharterBase$set_x_title(
  text,
  font_size = NULL,
  font_name = NULL,
  font_color = NULL,
  bold = NULL,
  italic = NULL,
  fill = NULL,
  line = NULL,
  line_width = NULL
)
```

Arguments:

`text` Title string.

`font_size` Numeric font size in points.

`font_name` Font typeface name.

`font_color` Six-digit hex color for the title text.

`bold` Logical.

`italic` Logical.

`fill` Six-digit hex color for the title background box.

`line` Six-digit hex color for the title border.

`line_width` Numeric border width in points.

Examples:

```
ec("line")$set_x_title("Month", font_color = "888888", italic = TRUE)
```

`EncharterBase$set_y_title()`: Set the primary Y-axis title.

Usage:

```
EncharterBase$set_y_title(
  text,
  font_size = NULL,
  font_name = NULL,
  font_color = NULL,
  bold = NULL,
  italic = NULL,
  fill = NULL,
  line = NULL,
  line_width = NULL
)
```

Arguments:

`text` Title string.

`font_size` Numeric font size in points.

`font_name` Font typeface name.

`font_color` Six-digit hex color for the title text.

`bold` Logical.

`italic` Logical.

`fill` Six-digit hex color for the title background box.

`line` Six-digit hex color for the title border.

`line_width` Numeric border width in points.

Examples:

```
ec("line")$set_y_title("Revenue (USD)", bold = TRUE)
```

`EncharterBase$set_x_axis()`: Set primary X-axis scaling, tick marks, and label formatting. Arguments that are not given keep their current value, so the method can be called several times.

Usage:

```
EncharterBase$set_x_axis(
  min = NULL,
  max = NULL,
  major = NULL,
  minor = NULL,
  major_time = NULL,
  minor_time = NULL,
  base_time = NULL,
  major_tick = NULL,
  minor_tick = NULL,
  format = NULL,
  log_base = NULL,
  rev = NULL,
  color = NULL,
  font_name = NULL,
  font_size = NULL,
  bold = NULL,
  italic = NULL,
  font_color = NULL,
  rotation = NULL,
  grid_color = NULL,
  grid_lines = NULL,
  minor_grid_color = NULL,
  minor_grid_lines = NULL,
  cross_between = NULL,
  line_width = NULL,
  grid_width = NULL,
  minor_grid_width = NULL,
  crosses = NULL,
  crosses_at = NULL,
```

```
  label_pos = NULL,
  tick_lbl_skip = NULL,
  tick_mark_skip = NULL,
  disp_units = NULL
)
```

Arguments:

`min, max` Numeric axis limits.

`major, minor` Numeric major/minor unit intervals. For date axes, unit is set by `major_time/minor_time`.

`major_time, minor_time` Time unit for major/minor steps on date axes: `"days"`, `"months"`, or `"years"`.

`base_time` Base time unit for date axes: `"days"`, `"months"`, or `"years"`.

`major_tick, minor_tick` Tick mark style: `"cross"`, `"in"`, `"out"`, or `"none"`.

`format` Number or date format string (e.g. `"#,##0"`, `"yyyy-mm-dd"`).

`log_base` Numeric base for logarithmic scaling (e.g. `10`).

`rev` Logical to reverse the value order

`color` Six-digit hex color for the axis line.

`font_name` Font typeface name for tick labels.

`font_size` Numeric label font size in points.

`bold, italic` Logical font style for tick labels.

`font_color` Six-digit hex color for axis tick labels. Defaults to `color` when not set.

`rotation` Numeric label rotation in degrees.

`grid_color, minor_grid_color` Six-digit hex colors for major/minor grid_lines.

`grid_lines, minor_grid_lines` Show grid lines. `TRUE/FALSE` to toggle; or a dash style string (`"dash"`, `"dot"`, `"dashDot"`, etc.) to show styled lines.

`cross_between` Where the value axis crosses: `"between"` (default, between categories) or `"midCat"` (through categories).

`line_width, grid_width, minor_grid_width` Numeric widths in points for the axis line, major grid lines, and minor grid lines respectively.

`crosses` Where this axis crosses its perpendicular axis: `"autoZero"` (default), `"min"`, or `"max"`.

`crosses_at` Numeric axis value at which to cross. Overrides `crosses` when supplied.

`label_pos` Tick label position: `"nextTo"` (default), `"high"`, `"low"`, or `"none"`.

`tick_lbl_skip` Integer (>= 1); draw a tick label only every n-th category (category axes only).

`tick_mark_skip` Integer (>= 1); draw a tick mark only every n-th category (category axes only).

`disp_units` Display units for value axes: a built-in unit string (`"hundreds"`, `"thousands"`, `"tenThousands"`, `"hundredThousands"`, `"millions"`, `"tenMillions"`, `"hundredMillions"`, `"billions"`, `"trillions"`) or a positive number for a custom unit.

Examples:

```
ec("line")$set_x_axis(
  min = 0, max = 12,
  major_tick = "out",
  grid_lines = TRUE,
  font_color = "666666",
  rotation   = -45
)
```

`EncharterBase$set_y_axis()`: Set primary Y-axis scaling, tick marks, and label formatting. Arguments that are not given keep their current value, so the method can be called several times.

Usage:

```
EncharterBase$set_y_axis(
  min = NULL,
  max = NULL,
  major = NULL,
  minor = NULL,
  major_time = NULL,
  minor_time = NULL,
  base_time = NULL,
  major_tick = NULL,
  minor_tick = NULL,
  format = NULL,
  log_base = NULL,
  rev = NULL,
  color = NULL,
  font_name = NULL,
  font_size = NULL,
  bold = NULL,
  italic = NULL,
  font_color = NULL,
  rotation = NULL,
  grid_color = NULL,
  grid_lines = NULL,
  minor_grid_color = NULL,
  minor_grid_lines = NULL,
  cross_between = NULL,
  line_width = NULL,
  grid_width = NULL,
  minor_grid_width = NULL,
  crosses = NULL,
  crosses_at = NULL,
  label_pos = NULL,
  tick_lbl_skip = NULL,
  tick_mark_skip = NULL,
  disp_units = NULL
)
```

Arguments:

`min`, `max` Numeric axis limits.

`major`, `minor` Numeric major/minor unit intervals.

`major_time`, `minor_time` Time unit for date axes: `"days"`, `"months"`, or `"years"`.

`base_time` Base time unit for date axes.

`major_tick`, `minor_tick` Tick mark style: `"cross"`, `"in"`, `"out"`, or `"none"`.

`format` Number format string.

`log_base` Numeric base for logarithmic scaling.

`rev` Logical to reverse the value order

`color` Six-digit hex color for the axis line.

`font_name` Font typeface name.

`font_size` Numeric label font size in points.

`bold`, `italic` Logical font style.

`font_color` Six-digit hex color for axis tick labels.

`rotation` Numeric label rotation in degrees.

`grid_color`, `minor_grid_color` Hex colors for major/minor grid lines.

`grid_lines`, `minor_grid_lines` `TRUE`/`FALSE` or a dash style string.

`cross_between` `"between"` or `"midCat"`.

`line_width`, `grid_width`, `minor_grid_width` Numeric widths in points.

`crosses` `"autoZero"`, `"min"`, or `"max"`.

`crosses_at` Numeric crossing value; overrides `crosses`.

`label_pos` `"nextTo"`, `"high"`, `"low"`, or `"none"`.

`tick_lbl_skip`, `tick_mark_skip` Integer (>= 1); label/tick every n-th category (category axes only).

`disp_units` Display units: a built-in unit string (e.g. `"thousands"`, `"millions"`) or a positive number for a custom unit. Value axes only.

Examples:

```
ec("bar")$set_y_axis(
  min        = 0,
  max        = 1000,
  major      = 200,
  format     = "#,##0",
  grid_lines = TRUE,
  grid_color = "DDDDDD"
)
```

`EncharterBase$set_data_label_style()`**:** Configure global data label defaults for all series. Per-series overrides can be set via the `show_val`/`show_cat` arguments in `$add_series()`.

Usage:

```
EncharterBase$set_data_label_style(
  show_val = TRUE,
  show_cat = FALSE,
  show_legend_key = FALSE,
  pos = "t",
  show_ser_name = FALSE,
  show_percent = FALSE,
  show_bubble_size = FALSE,
  format = NULL,
  ...
)
```

Arguments:

`show_val` Logical; show the data point value. Default `TRUE`.

`show_cat` Logical; show the category name. Default `FALSE`.

`show_legend_key` Logical; show the series color swatch next to each label. Default `FALSE`.

`pos` Label position: `"t"` (top, default), `"b"` (bottom), `"l"`, `"r"`, `"ctr"`, `"inEnd"`, `"outEnd"`, `"bestFit"`.

`show_ser_name` Logical; show the series name. Default `FALSE`.

`show_percent` Logical; show the percentage (pie-type charts). Default `FALSE`.

`show_bubble_size` Logical; show the bubble size (bubble charts). Default `FALSE`.

`format` Number format string for the labels (e.g. `"#,##0"`, `"0.0%"`).

`...` Additional font style arguments passed to the label text properties (e.g. `font_size`, `font_color`, `bold`).

Examples:

```
ec("bar")$set_data_label_style(show_val = TRUE, pos = "outEnd", font_size = 9)
```

`EncharterBase$set_legend_style()`: Configure the chart legend.

Usage:

```
EncharterBase$set_legend_style(
  pos = NULL,
  align = "ctr",
  overlay = FALSE,
  font_size = NULL,
  font_name = NULL,
  bold = NULL,
  italic = NULL,
  color = NULL
)
```

Arguments:

`pos` Legend position: `"t"`, `"b"`, `"l"`, `"r"`, or `"none"` to hide. Not given, the current position is kept: `"r"` for standard and `"t"` for extended charts, unless set before.

`align` Legend alignment relative to the chart: `"ctr"` (default), `"min"`, or `"max"`.

`overlay` Logical; if `TRUE` the legend overlaps the plot area.

`font_size` Numeric font size in points.

`font_name` Font typeface name.

`bold, italic` Logical font style.

`color` Six-digit hex color for the legend text.

Examples:

```
ec("line")$set_legend_style(pos = "b", font_size = 9)
```

`EncharterBase$set_chart_style()`: Style the outer chart area (background and border).

Usage:

```
EncharterBase$set_chart_style(fill = "FFFFFF", line = NULL, line_width = 1)
```

Arguments:

`fill` Six-digit hex color for the chart background. Default `"FFFFFF"`.

`line` Six-digit hex color for the chart border. `NULL` for no border.

`line_width` Numeric border width in points. Default `1`.

Examples:

```
ec("bar")$set_chart_style(fill = "F5F5F5", line = "CCCCCC", line_width = 0.5)
```

`EncharterBase$set_plot_style()`: Style the inner plot area (background and border).

Usage:

```
EncharterBase$set_plot_style(fill = NULL, line = NULL, line_width = 1)
```

Arguments:

`fill` Six-digit hex color for the plot area background. `NULL` for transparent.

`line` Six-digit hex color for the plot area border.

`line_width` Numeric border width in points. Default `1`.

Examples:

```
ec("line")$set_plot_style(fill = "FAFAFA")
```

`EncharterBase$print()`: Print a summary of the chart object.

Usage:

```
EncharterBase$print()
```

Examples:

```
ec("line")
```

`EncharterBase$clone()`: The objects of this class are cloneable with this method.

Usage:

```
EncharterBase$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

```
## ------------------------------------------------
## Method `EncharterBase$set_chart_title()`
## ------------------------------------------------

ec("line")$set_chart_title("Monthly Sales", font_size = 14, bold = TRUE)

## ------------------------------------------------
## Method `EncharterBase$set_x_title()`
## ------------------------------------------------

ec("line")$set_x_title("Month", font_color = "888888", italic = TRUE)

## ------------------------------------------------
## Method `EncharterBase$set_y_title()`
## ------------------------------------------------

ec("line")$set_y_title("Revenue (USD)", bold = TRUE)

## ------------------------------------------------
```

```
## Method `EncharterBase$set_x_axis()`
## ------------------------------------------------

ec("line")$set_x_axis(
  min = 0, max = 12,
  major_tick = "out",
  grid_lines = TRUE,
  font_color = "666666",
  rotation   = -45
)

## ------------------------------------------------
## Method `EncharterBase$set_y_axis()`
## ------------------------------------------------

ec("bar")$set_y_axis(
  min        = 0,
  max        = 1000,
  major      = 200,
  format     = "#,##0",
  grid_lines = TRUE,
  grid_color = "DDDDDD"
)

## ------------------------------------------------
## Method `EncharterBase$set_data_label_style()`
## ------------------------------------------------

ec("bar")$set_data_label_style(show_val = TRUE, pos = "outEnd", font_size = 9)

## ------------------------------------------------
## Method `EncharterBase$set_legend_style()`
## ------------------------------------------------

ec("line")$set_legend_style(pos = "b", font_size = 9)

## ------------------------------------------------
## Method `EncharterBase$set_chart_style()`
## ------------------------------------------------

ec("bar")$set_chart_style(fill = "F5F5F5", line = "CCCCCC", line_width = 0.5)

## ------------------------------------------------
## Method `EncharterBase$set_plot_style()`
## ------------------------------------------------

ec("line")$set_plot_style(fill = "FAFAFA")

## ------------------------------------------------
## Method `EncharterBase$print()`
## ------------------------------------------------

ec("line")
```

plot.Chart *Plot a chart*

Description

Draws a Chart or ChartEx object on the current graphics device with grid, approximating what a spreadsheet application shows for it.

Supported are bar/column (clustered, stacked, percent stacked, horizontal), line, area, scatter, bubble, pie, doughnut, pie of pie, bar of pie, radar and stock charts with titles, primary and secondary axes, gridlines, legend, markers, line styles, data labels, trendlines, error bars, high-low lines, drop lines and up-down bars; the 3D column, bar, line, area and pie types and surface charts (contour and 3D); and the extended types waterfall, box-and-whisker, histogram, Pareto, funnel, treemap and sunburst. Region maps are not drawn.

Usage

```
## S3 method for class 'Chart'
plot(x, wb = NULL, newpage = TRUE, ...)

## S3 method for class 'ChartEx'
plot(x, wb = NULL, newpage = TRUE, ...)
```

Arguments

x	A Chart or ChartEx object.
wb	Optional wbWorkbook the series reference.
newpage	Call grid::grid.newpage() first. Default TRUE.
...	Ignored.

Details

With wb, the values are read from the ranges of the series in that workbook, so one chart can be drawn for different workbooks. Cells in hidden rows and columns are skipped unless the chart plots hidden data (plot_vis_only). Without wb, the series caches (present when a series was added from wb_data()) are used. Series added with plain range strings and all ChartEx objects need wb.

Axis scaling follows the rules for automatic axes; fonts, spacing and the exact placement of labels differ from the original. Number formats are approximated for common codes (0, 0.00, #,##0, 0%, date formats).

Value

x, invisibly.

Examples

```
wb <- openxlsx2::wb_workbook()$add_worksheet("Data")$add_data(x = data.frame(
  Month = month.abb[1:6], Sales = c(120, 135, 128, 160, 175, 190)
))
chart <- ec("line")$set_chart_title("Sales")$
  add_series(name = Sales, data = openxlsx2::wb_data(wb), label = Month, marker = "circle")
plot(chart)
```

Index

Chart *(*encharter*)*, 2
ChartEx *(*encharter*)*, 2

ec *(*encharter*)*, 2
ec_from_crtx *(*encharter_crtx*)*, 20
ec_load *(*encharter_load*)*, 21
ec_to_crtx *(*encharter_crtx*)*, 20
encharter, 2
encharter(), *22*
encharter_crtx, 20
encharter_from_crtx *(*encharter_crtx*)*, 20
encharter_from_crtx(), *5*
encharter_load, 21
encharter_to_crtx *(*encharter_crtx*)*, 20
EncharterBase, *4*, *15*, 22

openxlsx2::wb_add_encharter(), *21*

plot.Chart, 32
plot.ChartEx *(*plot.Chart*)*, 32