Package: writexl (via r-universe)

August 21, 2026

Type Package

Title Export Data Frames to Excel 'xlsx' Format

Version 2.0.1

Description Zero-dependency data frame to xlsx exporter based on 'libxlsxwriter' <https://libxlsxwriter.github.io>. Fast and no Java or Excel required.

License BSD_2_clause + file LICENSE

Copyright Jeroen Ooms. For copyright notices of bundled third-party code (libxlsxwriter and dependencies), see COPYRIGHT.

Encoding UTF-8

URL https://ropensci.r-universe.dev/writexl

https://docs.ropensci.org/writexl/

BugReports https://github.com/ropensci/writexl/issues

Suggests grDevices, graphics, spelling, readxl, nycflights13, testthat (>= 3.2.0), xml2, bit64, knitr, rmarkdown

VignetteBuilder knitr

Language en-US

SystemRequirements zlib

Config/testthat/edition 3

Config/roxygen2/version 8.1.0

Config/pak/sysreqs zlib1g-dev

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

Date/Publication 2026-08-21 17:49:33 UTC

RemoteUrl https://github.com/ropensci/writexl

RemoteRef master

RemoteSha f4c283d996049da30d8704cbdddcf1e5579e2b49

Contents

is_xl_comment *Test whether an object is an 'xl_comment'*

Description

Test whether an object is an 'xl_comment'

Usage

```
is_xl_comment(x)
```

Arguments

x	An object.

Value

'TRUE' if 'x' inherits from '"xl_comment"'.

See Also

Other cell content: `xl_cell_general()`, `xl_comment()`, `xl_formula()`, `xl_rich_run()`, `xl_rich_string()`

is_xl_format *Test whether an object is an 'xl_format'*

Description

Test whether an object is an 'xl_format'

Usage

```
is_xl_format(x)
```

Arguments

x	An object.

Value

'TRUE' if 'x' inherits from '"xl_format"'.

See Also

Other cell formatting: `xl_color()`, `xl_format()`, `xl_format_groups`

lxw_version *Version*

Description

Shows version of bundled libxlsxwriter.

Usage

```
lxw_version()
```

write_xlsx *Export to xlsx*

Description

Writes a data frame to an xlsx file. To create an xlsx with (multiple) named sheets, simply set `x` to a named list of data frames.

Usage

```
write_xlsx(
  x,
  path = tempfile(fileext = ".xlsx"),
  col_names = TRUE,
  format_headers = TRUE,
  na = NA,
  use_zip64 = FALSE,
  constant_memory = NA,
  constant_memory_threshold = 128 * 1024^2
)
```

Arguments

`x`	a data frame, an [xl_sheet], an [xl_workbook], or a (named) list of data frames / 'xl_sheet's that become the sheets in the xlsx
`path`	a file name to write to
`col_names`	write column names as the header row at the top of the sheet?
`format_headers`	apply the workbook's header format to that header row? The default header format is bold and centered; change it with `xl_properties(header_format = )`.
`na`	what to write where a value has none. 'NA' (the default) leaves the cell blank, as writexl has always done; anything else is written in its place, keeping its own type. Shorthand for `xl_properties(na = )`, so give it there instead when 'x' is already an [xl_workbook]. A column or a single cell can override it: see [xl_col_spec()] and [xl_cell_general()].

`use_zip64` use zip64 to enable support for 4GB+ xlsx files. Not all platforms can read this.

`constant_memory`
: stream rows to disk instead of building the whole workbook in memory. 'NA' (the default) decides per workbook: on for large data, off for small, and always off when a feature needs it off. 'TRUE' forces it on for a workbook that would otherwise be judged too small; 'FALSE' forces it off. Features that cannot be written while streaming — merged ranges, tables, embedded images and multi-cell array formulas — turn it off regardless, with a warning if 'TRUE' was asked for, because the alternative is a file that opens cleanly and is missing cells.

`constant_memory_threshold`
: how much extra memory not streaming would have to cost, in bytes, before streaming is worth it. Default 128 MiB. The cost is *estimated* from the number of cells in the workbook, using a fixed per-cell figure calibrated against a range of data; the true cost varies with the data, and is lowest for text that repeats. Streaming saves memory but produces slightly larger files, so it is not used for workbooks small enough that the saving would not be noticed.

Details

Supports strings, numbers, booleans and dates automatically. For cell formatting (fonts, fills, borders, number formats, ...), worksheet layout (column widths, frozen panes, ...), and workbook metadata, wrap columns with `xl_cell_general`, sheets with `xl_sheet`, and the whole workbook with `xl_workbook`. See the "Formatting and workbook properties" vignette and `xl_format`.

See Also

Other workbook settings: `xl_properties()`, `xl_workbook()`

Examples

```
# Roundtrip example with single excel sheet named 'mysheet'
tmp <- write_xlsx(list(mysheet = iris))
readxl::read_xlsx(tmp)
```

---

xl_cell_general *General cell objects for Excel writing*

Description

'xl_cell_general' creates a vector of cell objects, each optionally containing a **value**, a **formula**, and/or a **hyperlink**. It is the fundamental building block used internally by [xl_formula()] and [xl_hyperlink()], and can be used directly for mixed-type columns or cells that combine multiple features (e.g., a formula with a pre-calculated result, or a URL with separate display text and tooltip).

An 'xl_cell_general' behaves like a vector: it has a 'length()', supports '[', '[[', 'c()' and 'rep()', and recycles automatically when assigned to a data frame column of a different length (just like

[xl_formula()]). It is deliberately not a list, so that ‘df[, j] <- cells‘ assigns one column rather than being read as a list of them.

‘df[i, j] <- x‘ sets that cell’s ‘value‘, whatever ‘x‘’s type; a formula needs [xl_formula()], since a cell column carries no column-wide notion of "these are all formulas".

‘as.character()‘ returns what each cell displays, so a cell built for a sheet can be reused wherever a plain string is wanted — [xl_merge()]’s or [xl_comment()]’s ‘value‘, for instance. A cell that carries only a formula shows ‘NA‘: its displayed value comes from Excel, not from writexl.

Usage

```
xl_cell_general(
  value = NULL,
  formula = NULL,
  hyperlink = NULL,
  format = NULL,
  comment = NULL,
  array = FALSE,
  dynamic = FALSE,
  array_range = NULL,
  na = NA
)

## S3 method for class 'xl_cell_general'
as.character(x, ...)
```

Arguments

value	An atomic vector or a list of scalars, one per cell. Use ‘NA‘ for a cell that is empty unless ‘na‘ says otherwise (see ‘na‘ below, and note that a workbook-wide ‘na‘ reaches these values too). A list enables mixed types across cells in the same column (e.g., ‘list(1.5, "text", TRUE)‘). Date and POSIXct scalars are supported and formatted as in [write_xlsx()]. When ‘hyperlink‘ is also set for the same cell, a **character** ‘value‘ is used as the display text shown in the cell instead of the raw URL; all other types are ignored for hyperlink cells.
formula	A character vector of Excel formulas (each must start with ‘"="‘), or ‘NA‘ for cells with no formula. When both ‘value‘ and ‘formula‘ are supplied for the same cell, ‘value‘ is used as a pre-calculated result stored alongside the formula via ‘worksheet_write_formula_num()‘ (numeric value) or ‘worksheet_write_formula_str()‘ (character value). This allows static xlsx exports that display formula text in the formula bar but do not require Excel to recalculate on open.
hyperlink	A character vector of URLs, or a list where each element is ‘NA‘, a single character URL, or a named list with elements: **‘url‘** (required) The target URL. **‘tooltip‘** (optional) Tooltip text shown on hover. Supply a character ‘value‘ alongside ‘hyperlink‘ to show custom display text in the cell instead of the raw URL. The hyperlink is written via ‘worksheet_write_url_opt()‘.

`format` An [xl_format] object (applied to every cell), or a list of 'xl_format' objects (one per cell, recycled), or 'NULL' for no formatting. Build formats with [xl_font()], [xl_fill()], [xl_border()], [xl_align()], [xl_num_format()] and [xl_protection()], combined with [xl_format()] or '+'. When a cell's value is a date/time and its format sets no number format, the default date/time number format is applied automatically.

`comment` Cell comments (notes): a character vector of comment text (one per cell, recycled; 'NA' for no comment), a single [xl_comment()] (recycled to every cell), or a list mixing strings / 'xl_comment' / 'NA' per cell. 'NULL' for no comments.

`array, dynamic` Logical (one per cell, recycled): how the cell's 'formula' is stored. 'array = TRUE' writes a legacy *array* (Ctrl-Shift- Enter) formula; 'dynamic = TRUE' writes a modern *dynamic array* formula, which Excel spills over as many cells as the result needs. Neither can carry a character 'value', because Excel stores no cached string result for an array formula. On a cell that has no 'formula' the flags are inert, so a single 'array = TRUE' can be recycled across a column that mixes formula and value cells.

`array_range` The range a legacy array formula covers, for the rare case where it must be declared: an Excel range string ('"C2:C11"') or a 'list(rows = , cols =)' spec, one per cell ('NA' for none). It must start at the cell holding the formula, and must extend into cells the sheet does not otherwise write — the range is padded on write, so an overlap would have the padding and the sheet's own values overwrite each other.

Leave it unset for almost everything: a single-cell 'dynamic' formula spills automatically in Excel 365 / 2021, and a single-cell 'array' formula is the right spelling for a 'SUMPRODUCT'-style aggregate. Supplying it forces the workbook out of the memory-efficient row-streaming mode, since libxlsxwriter cannot pad an array range while streaming.

`na` What to write in a cell that has no value, one per cell and recycled. 'NA' (the default) inherits the column's [xl_col_spec()]'(na =)', and failing that the workbook's [xl_properties()]'(na =)'; a cell that sets its own overrides both. This is also how a cell asks for a blank where the column or workbook would otherwise substitute something — give it the value you want, '""' for an empty string.

`x` An 'xl_cell_general'.

`...` Ignored.

Value

An object of class 'c("xl_cell_general", "xl_cell")', which is a list of length 'n' where each element is a named list with fields 'value', 'formula', 'hyperlink', 'format', 'comment', 'array', 'dynamic' and 'array_range'.

See Also

[xl_formula()], [xl_hyperlink()], [write_xlsx()]

Other cell content: `is_xl_comment()`, `xl_comment()`, `xl_formula()`, `xl_rich_run()`, `xl_rich_string()`

Examples

```
# Value-only cell
xl_cell_general(value = 42)

# Formula with a pre-calculated numeric result (static export)
xl_cell_general(value = 42.0, formula = "=SUM(A1:A10)")

# Hyperlink with display text (value) and tooltip
xl_cell_general(
  value    = "Visit",
  hyperlink = list(url = "https://example.com", tooltip = "Go to example.com")
)

# Vector of cells: value and formula cells in one column
cells <- c(
  xl_cell_general(value = 1.5),
  xl_cell_general(value = "note"),
  xl_cell_general(formula = "=A1+A2")
)

# Used in a data frame (length-1 recycles to fill all rows, as with
# xl_formula())
df <- data.frame(x = 1:3)
df$formula_col <- xl_formula("=A1*2")   # backward-compatible shorthand
df$cell_col    <- xl_cell_general(value = 99L)  # all rows get 99
```

---

xl_chart *Add a chart to a worksheet*

Description

'xl_chart()' builds a chart from one or more [xl_chart_series()] and places it on a sheet, anchored to a cell. Pass one or a list of them as 'xl_sheet(chart =)'.

Placement works exactly as it does for [xl_image()] — 'at', 'scale', 'offset', 'position', 'description' and 'decorative' mean the same things, because libxlsxwriter describes both with the same fields.

Usage

```
xl_chart(
  type,
  series,
  title = NULL,
  title_format = NULL,
  title_layout = NULL,
  title_overlay = NA,
  x_axis = NULL,
  y_axis = NULL,
  legend = NULL,
```

```
  data_table = NULL,
  plot_area_format = NULL,
  plot_area_layout = NULL,
  chart_area_format = NULL,
  drop_lines = NA,
  high_low_lines = NA,
  up_down_bars = NA,
  hole_size = NA,
  rotation = NA,
  series_gap = NA,
  series_overlap = NA,
  show_blanks = NULL,
  show_hidden_data = NA,
  at = "A1",
  scale = 1,
  offset = NULL,
  position = "move_and_size",
  description = NULL,
  decorative = FALSE,
  style = NA
)
```

Arguments

`type`	The chart type: ‘"column"‘, ‘"bar"‘, ‘"line"‘, ‘"pie"‘, ‘"doughnut"‘, ‘"area"‘, ‘"scatter"‘, ‘"radar"‘, and the stacked, percent-stacked, smoothed and marker variants.
`series`	One [xl_chart_series()], or a list of them. Every series of a scatter chart must have ‘categories‘, which are its x axis.
`title`	The chart title. A string is always taken literally, so to take the title from a cell give a range spec — ‘list(header = "revenue")‘ for a column’s header cell, or ‘list(rows = 1, cols = 1)‘ for a data cell. ‘FALSE‘ removes the title Excel would otherwise generate.
`title_format`	An [xl_format()] styling the title text. A title is text, so only the [xl_font()] group applies.
`title_layout`	Where to put the title by hand, as ‘c(x, y)‘ fractions of the chart. Excel places it for you otherwise.
`title_overlay`	Let the title sit over the plot rather than above it.
`x_axis`, `y_axis`	An [xl_chart_axis()] describing that axis. Pie and doughnut charts have none, and several axis options apply to a value or a category axis only — see [xl_chart_axis()].
`legend`	An [xl_chart_legend()] moving, styling or removing the legend.
`data_table`	An [xl_chart_table()] printing the plotted numbers in a grid beneath the chart.
`plot_area_format`, `chart_area_format`	An [xl_format()] styling the plot area — the panel the data is drawn in — and the chart area around it: [xl_border()] for the line, [xl_fill()] for the fill or pattern.

`plot_area_layout`
: Where to put the plot area by hand, as ‘c(x, y)‘ or ‘c(x, y, width, height)‘ fractions of the chart.

`drop_lines`
: Drop lines from each point to the category axis: ‘TRUE‘, or an [xl_format()] giving the line to draw them with. Line and area charts.

`high_low_lines`
: A line joining the highest and lowest series at each category, the same way. Line charts.

`up_down_bars`
: Bars between the first and last series at each category: ‘TRUE‘, or ‘list(up = , down =)‘ with an [xl_format()] for either bar. Line charts.

`hole_size`
: The size of a doughnut’s hole, 10 to 90 percent.

`rotation`
: Where a pie or doughnut starts, 0 to 360 degrees clockwise from the top.

`series_gap`
: The gap between category groups on a bar or column chart, 0 to 500 percent of a bar’s width.

`series_overlap`
: How far bars of one category overlap, -100 to 100 percent. 100 stacks them, -100 pushes them apart.

`show_blanks`
: What an empty cell does to the plot: leave a ‘"gap"‘, plot it as ‘"zero"‘, or join across it with ‘"connected"‘.

`show_hidden_data`
: Plot data from rows and columns that are hidden. Excel leaves them out otherwise.

`at`
: The cell the chart’s top-left corner is anchored to.

`scale`
: Scale factor: one number for both axes, or ‘c(x, y)‘.

`offset`
: Offset from the anchor cell’s corner in pixels, as ‘c(x, y)‘.

`position`
: How the chart behaves when rows and columns change size; see [xl_image()].

`description`
: Alt text, for screen readers.

`decorative`
: Mark the chart as decorative, so screen readers skip it.

`style`
: Excel’s built-in chart style, 1–48.

Value

An ‘xl_chart‘ object.

What a chart type supports

Excel silently drops options a chart type cannot use, so writexl refuses them instead, naming the types that would work. Pie and doughnut charts have no axes; only a doughnut has a hole; only pie and doughnut rotate; up-down bars and high-low lines are line-only; the series gap and overlap are bar and column only.

See Also

[xl_chart_series], [xl_sheet]

Other images and charts: `xl_chart_axis()`, `xl_chart_error_bars()`, `xl_chart_labels()`, `xl_chart_legend()`, `xl_chart_marker()`, `xl_chart_series()`, `xl_chart_table()`, `xl_chart_trendline()`, `xl_chartsheet()`, `xl_image()`

Examples

```
xl_chart("column", xl_chart_series(values = list(cols = "revenue")))
xl_chart("pie", xl_chart_series(values = "Data!B2:B5"), title = "Share")
```

`xl_chart_axis` *An axis of a chart*

Description

'xl_chart_axis()' describes one axis, and is given to [xl_chart()] as 'x_axis' or 'y_axis'.

Several options apply to one kind of axis only, and Excel discards the rest without a word, so writexl refuses them instead. A scatter chart plots numbers against numbers, so both of its axes are **value** axes; every other type has a **category** x axis and a value y axis. (A bar chart is drawn with its categories up the side, but the axes keep their names.) Pie and doughnut charts have no axes at all.

* value axes only — 'min', 'max', 'log_base', 'major_unit', 'minor_unit', 'display_units', 'display_units_visible'; * category axes only — 'position', 'label_align', 'interval_unit', 'interval_tick'.

Usage

```
xl_chart_axis(
  title = NULL,
  title_format = NULL,
  title_layout = NULL,
  label_format = NULL,
  num_format = NULL,
  line_format = NULL,
  visible = NA,
  reverse = NA,
  min = NA,
  max = NA,
  log_base = NA,
  major_unit = NA,
  minor_unit = NA,
  display_units = NULL,
  display_units_visible = NA,
  interval_unit = NA,
  interval_tick = NA,
  position = NULL,
  label_position = NULL,
  label_align = NULL,
  major_tick = NULL,
  minor_tick = NULL,
  crossing = NULL,
  major_gridlines = NA,
  minor_gridlines = NA,
```

```
  major_gridlines_format = NULL,
  minor_gridlines_format = NULL
)
```

Arguments

`title`	The axis title: a string, or a range spec holding one — see [xl_chart_series()] for the spellings, including ‘list(header = "revenue")‘.
`title_format`	An [xl_format()] styling the axis title. A title is text, so only the [xl_font()] group applies.
`title_layout`	Where to put the axis title by hand, as ‘c(x, y)‘ fractions of the chart, each above 0 and at most 1. Excel places it for you otherwise.
`label_format`	An [xl_format()] styling the tick labels — the [xl_font()] group only.
`num_format`	A number format for the tick labels, as an Excel format string (‘"#,##0"‘) or an [xl_num_format()].
`line_format`	An [xl_format()] styling the axis line itself: [xl_border()] for the line, [xl_fill()] for the fill behind it. An axis has four parts that can be styled, so none of them is just ‘format‘.
`visible`	‘FALSE‘ hides the axis.
`reverse`	Draw the axis in the opposite direction.
`min`, `max`	The axis bounds. Value axes only.
`log_base`	Use a logarithmic scale with this base, 2 or more. Value axes only.
`major_unit`, `minor_unit`	The spacing between major and minor tick marks. Value axes only.
`display_units`	Scale the labels by ‘"thousands"‘, ‘"millions"‘, ‘"billions"‘ and so on; see Details for the full set. Value axes only.
`display_units_visible`	Whether the caption naming the units — the small rotated "Millions" beside the axis — is drawn. Setting ‘display_units‘ turns it **on**, as Excel does, so this is really for ‘FALSE‘: rescaled labels with no caption. Value axes only.
`interval_unit`	Label one category in every ‘n‘. Category axes only.
`interval_tick`	Put a tick mark on one category in every ‘n‘. Category axes only.
`position`	Whether the data sits ‘"on_tick"‘ or ‘"between"‘ the tick marks. Category axes only.
`label_position`	Where the tick labels go: ‘"next_to"‘, ‘"high"‘, ‘"low"‘, or ‘"none"‘ for no labels.
`label_align`	Tick-label alignment: ‘"center"‘, ‘"left"‘ or ‘"right"‘. Category axes only.
`major_tick`, `minor_tick`	The tick marks: ‘"default"‘, ‘"none"‘, ‘"inside"‘, ‘"outside"‘ or ‘"crossing"‘.
`crossing`	Where the other axis crosses this one: a number, or ‘"min"‘ or ‘"max"‘ for either end.
`major_gridlines`, `minor_gridlines`	Show the gridlines. A chart’s major y gridlines are on by default and everything else is off.

`major_gridlines_format`, `minor_gridlines_format`
: An [xl_format()] styling the gridlines — [xl_border()] only, since a gridline is a line.

Details

'display_units' is one of '"none"', '"hundreds"', '"thousands"', '"ten_thousands"', '"hundred_thousands"', '"millions"', '"ten_millions"', '"hundred_millions"', '"billions"' or '"trillions"'.

Value

An 'xl_chart_axis' object.

See Also

[xl_chart], [xl_chart_series]

Other images and charts: `xl_chart()`, `xl_chart_error_bars()`, `xl_chart_labels()`, `xl_chart_legend()`, `xl_chart_marker()`, `xl_chart_series()`, `xl_chart_table()`, `xl_chart_trendline()`, `xl_chartsheet()`, `xl_image()`

Examples

```
xl_chart_axis(title = "Quarter")
xl_chart_axis(title = "Revenue", min = 0, num_format = "$#,##0",
              major_gridlines = FALSE)
```

---

xl_chart_error_bars *Error bars on a chart series*

Description

'xl_chart_error_bars()' draws an error bar at each point, and is given to [xl_chart_series()] as 'x_error_bars' or 'y_error_bars'.

Usage

```
xl_chart_error_bars(
  type,
  value = NA,
  direction = NULL,
  endcap = NA,
  format = NULL
)
```

Arguments

type	How the size of each bar is worked out: ‘"std_error"‘ for the standard error, ‘"fixed"‘ for a constant, ‘"percentage"‘ of the point’s own value, or ‘"std_dev"‘ for that many standard deviations.
value	The constant, the percentage, or the number of standard deviations. ‘"std_error"‘ needs none.
direction	‘"both"‘, ‘"plus"‘ or ‘"minus"‘.
endcap	Draw the cap at the end of each bar. On by default.
format	An [xl_format()] styling the bars — [xl_border()] only, since an error bar is a line.

Value

An ‘xl_chart_error_bars‘ object.

See Also

[xl_chart_series]

Other images and charts: xl_chart(), xl_chart_axis(), xl_chart_labels(), xl_chart_legend(), xl_chart_marker(), xl_chart_series(), xl_chart_table(), xl_chart_trendline(), xl_chartsheet(), xl_image()

Examples

```
xl_chart_error_bars("percentage", 5)
xl_chart_error_bars("std_dev", 1, direction = "plus", endcap = FALSE)
```

---

xl_chart_labels *Data labels on a chart series*

Description

‘xl_chart_labels()‘ prints the numbers next to the points that carry them. With no arguments it shows the value of each point, which is Excel’s own default.

‘xl_chart_label()‘ describes one label, for ‘custom‘: a label can be given its own text or hidden, one point at a time.

Excel allows different label positions for different chart types, and drops one that does not apply. The table is in libxlsxwriter’s header and is enforced here: ‘"center"‘ is allowed everywhere, ‘"right"‘, ‘"left"‘, ‘"above"‘ and ‘"below"‘ on line and scatter charts, ‘"inside_base"‘ on bar and column, ‘"inside_end"‘ and ‘"outside_end"‘ on bar, column, pie and doughnut, and ‘"best_fit"‘ on pie and doughnut.

Usage

```
xl_chart_labels(
  show_value = NA,
  show_name = NA,
  show_category = NA,
  show_percentage = NA,
  show_legend_key = NA,
  num_format = NULL,
  position = NULL,
  separator = NULL,
  format = NULL,
  leader_lines = NA,
  custom = NULL
)

xl_chart_label(value = NULL, hide = NA, format = NULL)
```

Arguments

show_value, show_name, show_category, show_percentage	What each label holds: the point's value, the series name, the category, and the value as a percentage of the series. Naming any of them means the label holds exactly those, so 'show_percentage = TRUE' alone gives a percentage and nothing else; naming none of them leaves Excel's default, the value. Percentages are meaningful on pie and doughnut charts.
show_legend_key	Print the series' legend swatch in each label.
num_format	A number format for the labels, as an Excel format string or an [xl_num_format()].
position	Where the label sits relative to its point; see the description for which chart types allow which.
separator	What joins the parts of a label when it holds more than one: '"comma"', '"semicolon"', '"period"', '"newline"' or '"space"'.
format	An [xl_format()] styling the labels. A label is a shape with text in it, so all of [xl_font()], [xl_border()] and [xl_fill()] apply.
leader_lines	Draw a line from a label back to its point. Excel only shows one once the label has been dragged away from the point.
custom	A list of [xl_chart_label()]s, one per point in order, giving individual labels their own text, styling, or 'hide = TRUE'. 'NULL' in the list leaves that point's label alone.
value	The label's text. A string beginning with '"="' is a formula, so '"=Sheet1!A1"' takes the text from a cell.
hide	Remove this point's label, leaving the others.

Value

An 'xl_chart_labels' object.

See Also

[xl_chart_series]

Other images and charts: xl_chart(), xl_chart_axis(), xl_chart_error_bars(), xl_chart_legend(), xl_chart_marker(), xl_chart_series(), xl_chart_table(), xl_chart_trendline(), xl_chartsheet(), xl_image()

Examples

```
xl_chart_labels()
xl_chart_labels(show_category = TRUE, show_percentage = TRUE,
                separator = "newline", position = "outside_end")
```

---

xl_chart_legend *A chart's legend*

Description

‘xl_chart_legend()‘ moves, styles or removes the legend, and can leave individual series out of it.

Usage

```
xl_chart_legend(
  position = NULL,
  format = NULL,
  layout = NULL,
  delete_series = NULL
)
```

Arguments

position	Where the legend sits: ‘"right"‘ (Excel's default), ‘"left"‘, ‘"top"‘, ‘"bottom"‘, ‘"top_right"‘, the ‘"overlay_*"‘ variants that let the legend sit over the plot, or ‘"none"‘ to remove it.
format	An [xl_format()] styling the legend text — the [xl_font()] group only, since libxlsxwriter gives a legend a font and nothing else.
layout	Where to put the legend by hand, as ‘c(x, y)‘ or ‘c(x, y, width, height)‘ — fractions of the chart, each above 0 and at most 1. Excel places it for you otherwise. ‘at‘ is a cell everywhere else in writexl, so a chart's own fractions are a ‘layout‘.
delete_series	Series to leave out of the legend, by position: ‘2‘ drops the second series' entry while still plotting it. This is how a trendline or a helper series is kept out of the key.

Value

An ‘xl_chart_legend‘ object.

See Also

[xl_chart]

Other images and charts: xl_chart(), xl_chart_axis(), xl_chart_error_bars(), xl_chart_labels(), xl_chart_marker(), xl_chart_series(), xl_chart_table(), xl_chart_trendline(), xl_chartsheet(), xl_image()

Examples

```
xl_chart_legend(position = "bottom")
xl_chart_legend(position = "none")
xl_chart_legend(delete_series = 2)
```

---

xl_chart_marker *A marker on a chart series*

Description

‘xl_chart_marker()‘ draws a symbol at each point of a series. Line, scatter and radar charts are where they show; on other types Excel ignores them.

‘type = "automatic"‘ asks Excel for the default marker of that series, and is the one type that cannot be given a size or a format — libxlsxwriter documents that, and Excel drops them, so both are refused here.

Usage

```
xl_chart_marker(type = NULL, size = NA, format = NULL)
```

Arguments

type	The symbol: ‘"automatic"‘, ‘"none"‘, ‘"square"‘, ‘"diamond"‘, ‘"triangle"‘, ‘"x"‘, ‘"star"‘, ‘"short_dash"‘, ‘"long_dash"‘, ‘"circle"‘ or ‘"plus"‘.
size	The symbol’s size in points, 2 to 72.
format	An [xl_format()] styling the symbol: [xl_border()] for its outline, [xl_fill()] for its fill or pattern.

Value

An ‘xl_chart_marker‘ object.

See Also

[xl_chart_series]

Other images and charts: xl_chart(), xl_chart_axis(), xl_chart_error_bars(), xl_chart_labels(), xl_chart_legend(), xl_chart_series(), xl_chart_table(), xl_chart_trendline(), xl_chartsheet(), xl_image()

Examples

```
xl_chart_marker(type = "circle", size = 8)
xl_chart_marker(type = "none")
```

`xl_chart_series` *A data series within a chart*

Description

‘xl_chart_series()‘ names the values a chart plots, and optionally the categories to plot them against and a name for the legend. A series that plots a column is named after that column’s header unless told otherwise.

Each range may live on a different sheet from the chart, so it takes an optional ‘sheet‘:

* ‘"Data!B2:B10"‘ — an A1 range, sheet-qualified; * ‘list(cols = "revenue")‘ — resolved against the chart’s own sheet; * ‘list(sheet = "Data", cols = "revenue")‘ — against another sheet; * ‘list(header = "revenue")‘ — that column’s header cell, which is where a series name usually lives.

A range that selects no data is an error rather than an empty chart.

Usage

```
xl_chart_series(
  values,
  categories = NULL,
  name = NULL,
  format = NULL,
  marker = NULL,
  labels = NULL,
  trendline = NULL,
  x_error_bars = NULL,
  y_error_bars = NULL,
  points = NULL,
  smooth = NA,
  invert_if_negative = NA
)
```

Arguments

`values`	The range holding the numbers to plot.
`categories`	The range holding the labels to plot them against. Omit for a chart that numbers its points.
`name`	The series name, shown in the legend. Left unset, a series that plots a column takes its name from that column’s header cell, which is what Excel does when you chart a column along with its header; ‘FALSE‘ leaves it unnamed. A string is always taken literally — a series may legitimately be called ‘"Q1!"‘ — so to take the name from another cell, give a range spec: ‘name = list(header = "cost")‘ for a different column’s header, or ‘name = list(rows = 1, cols = 1)‘ for a data cell.

`format` An [xl_format] styling the series — its line and fill. See [xl_chart()] for which format properties a chart can express.

`marker` An [xl_chart_marker()] drawn at each point.

`labels` An [xl_chart_labels()] printing the numbers beside the points.

`trendline` An [xl_chart_trendline()] fitted through the series.

`x_error_bars, y_error_bars` An [xl_chart_error_bars()] on each point.

`points` An [xl_format()] per point, as a list, styling individual points — one slice of a pie, one bar of a column chart. 'NULL' in the list leaves that point as it is.

`smooth` Draw the line smoothed. Line and scatter charts only.

`invert_if_negative` Fill negative values with the inverse colour.

Value

An 'xl_chart_series' object.

See Also

[xl_chart]

Other images and charts: `xl_chart()`, `xl_chart_axis()`, `xl_chart_error_bars()`, `xl_chart_labels()`, `xl_chart_legend()`, `xl_chart_marker()`, `xl_chart_table()`, `xl_chart_trendline()`, `xl_chartsheet()`, `xl_image()`

Examples

```
xl_chart_series(values = list(cols = "revenue"))
xl_chart_series(values = "Data!B2:B10", categories = "Data!A2:A10",
                name = "2024")
```

---

xl_chart_table *The table of values under a chart*

Description

'xl_chart_table()' prints the plotted numbers in a grid beneath the chart, which is Excel's "Data Table" chart element. It is given to [xl_chart()] as 'data_table'.

Naming none of the grid options leaves Excel's own: horizontal, vertical and outline borders drawn, and no legend keys.

Usage

```
xl_chart_table(
  show_keys = NA,
  horizontal_border = NA,
  vertical_border = NA,
  outline_border = NA,
  format = NULL
)
```

Arguments

show_keys	Print each series' legend swatch in the table.
horizontal_border, vertical_border, outline_border	Which of the grid's borders to draw.
format	An [xl_format()] styling the table's text — the [xl_font()] group only.

Value

An 'xl_chart_table' object.

See Also

[xl_chart]

Other images and charts: xl_chart(), xl_chart_axis(), xl_chart_error_bars(), xl_chart_labels(), xl_chart_legend(), xl_chart_marker(), xl_chart_series(), xl_chart_trendline(), xl_chartsheet(), xl_image()

Examples

```
xl_chart_table()
xl_chart_table(show_keys = TRUE, vertical_border = FALSE)
```

---

xl_chart_trendline *A trendline on a chart series*

Description

'xl_chart_trendline()' fits a line through a series.

Two of Excel's own restrictions are enforced, because it discards these rather than complain: a **moving average** has no forecast, no equation and no R-squared, and an **intercept** applies only to exponential, linear and polynomial fits.

Usage

```
xl_chart_trendline(
  type,
  order = NA,
  period = NA,
  forward = NA,
  backward = NA,
  intercept = NA,
  equation = NA,
  r_squared = NA,
  name = NULL,
  format = NULL
)
```

Arguments

type	'"linear"', '"log"', '"poly"', '"power"', '"exp"' or '"average"' for a moving average.
order	The order of a polynomial fit, 2 or more. '"poly"' only.
period	The number of points a moving average covers, 2 or more. '"average"' only.
forward, backward	How far to project the line beyond the data, in categories.
intercept	Force the line through this value on the y axis. Exponential, linear and polynomial fits only.
equation	Print the fitted equation on the chart.
r_squared	Print the R-squared value on the chart.
name	The trendline's name in the legend. Excel generates one otherwise.
format	An [xl_format()] styling the line — [xl_border()] only, since a trendline is a line.

Value

An 'xl_chart_trendline' object.

See Also

[xl_chart_series]

Other images and charts: xl_chart(), xl_chart_axis(), xl_chart_error_bars(), xl_chart_labels(), xl_chart_legend(), xl_chart_marker(), xl_chart_series(), xl_chart_table(), xl_chartsheet(), xl_image()

Examples

```
xl_chart_trendline("linear", equation = TRUE, r_squared = TRUE)
xl_chart_trendline("poly", order = 3)
xl_chart_trendline("average", period = 2)
```

`xl_chartsheet` *A sheet holding a single chart*

Description

‘xl_chartsheet()‘ is a worksheet-sized chart: a tab of its own holding one chart and no cells. Give it to [write_xlsx()] in place of a data frame.

Because a chartsheet has no cells, every range in the chart’s series must name the sheet it plots — ‘list(sheet = "Data", cols = "revenue")‘ or ‘"Data!B2:B10"‘. A bare ‘list(cols =)‘ has nothing to resolve against and is refused.

A chartsheet supports only part of what a worksheet does, and the parts it does not are refused rather than dropped: of [xl_page_setup()] it takes the orientation, paper size, margins and the header and footer; of [xl_sheet_view()] it takes ‘active‘, ‘selected‘, ‘visible‘ and ‘first_tab‘.

Usage

```
xl_chartsheet(
  chart,
  tab_color = NULL,
  zoom = NA,
  protect = NULL,
  page = NULL,
  view = NULL
)
```

Arguments

`chart`	The [xl_chart()] to fill the sheet with.
`tab_color`	The colour of the sheet tab.
`zoom`	The zoom level as a percentage, 10 to 400.
`protect`	‘TRUE‘, a password string, or a named list. A chartsheet has no cells, so of Excel’s protection options it takes only ‘no_content‘ (let the chart be edited) and ‘no_objects‘ (let the shapes on it be edited); the worksheet options are refused by name.
`page`	An [xl_page_setup()] describing how it prints.
`view`	An [xl_sheet_view()] setting the tab state.

Value

An ‘xl_chartsheet‘ object.

See Also

[xl_chart], [xl_sheet]

Other images and charts: `xl_chart()`, `xl_chart_axis()`, `xl_chart_error_bars()`, `xl_chart_labels()`, `xl_chart_legend()`, `xl_chart_marker()`, `xl_chart_series()`, `xl_chart_table()`, `xl_chart_trendline()`, `xl_image()`

Examples

```
sales <- data.frame(quarter = c("Q1", "Q2"), revenue = c(10, 25))
chart <- xl_chart("column",
                  xl_chart_series(values = list(sheet = "Data",
                                                cols = "revenue")))
write_xlsx(list(Data = sales, Overview = xl_chartsheet(chart)),
           tempfile(fileext = ".xlsx"))
```

`xl_color` *Normalize a color to a libxlsxwriter RGB integer*

Description

Converts an R color name (e.g. ‘"navy"‘), a hex string (‘"#FF0000"‘ or ‘"FF0000"‘), or an integer in the range ‘0x000000‘..‘0xFFFFFF‘ into the single ‘0xRRGGBB‘ integer that libxlsxwriter expects. Used internally by every ‘color‘/‘background‘/‘foreground‘ argument of the formatting constructors, and exported so it can be used directly.

Usage

```
xl_color(x)
```

Arguments

`x`	A single color: an R color name, a hex string, or an integer. ‘NA‘ returns ‘NA_integer_‘ (an unset color).

Value

A single integer in ‘0x000000‘..‘0xFFFFFF‘, or ‘NA_integer_‘.

See Also

Other cell formatting: `is_xl_format()`, `xl_format()`, `xl_format_groups`

Examples

```
xl_color("red")
xl_color("#0000FF")
xl_color(255L)
```

xl_colrow_spec *Column and row specifications for a worksheet*

Description

'xl_col_spec()' and 'xl_row_spec()' describe formatting and geometry for a set of columns or rows within a sheet built by [xl_sheet()]. They are *subclasses* of [xl_format]: they carry the usual formatting groups (so they combine with '+' and the group constructors) plus a target (which columns/rows) and geometry (width/height, hidden, outline level).

Usage

```
xl_col_spec(
  cols,
  width = NA,
  hidden = NA,
  level = NA,
  format = NULL,
  width_pixels = NA,
  collapsed = NA,
  na = NA
)

xl_row_spec(
  rows,
  height = NA,
  hidden = NA,
  level = NA,
  format = NULL,
  height_pixels = NA,
  collapsed = NA
)
```

Arguments

cols	Columns to target: a character vector of column names or a numeric vector of 1-based positions.
width	Column width (in Excel character units).
hidden	Logical; hide the column/row.
level	Integer outline (grouping) level, 0–7.
format	An optional [xl_format] applied to the column/row as its default cell format. Combine groups with '+' (e.g. 'xl_font(bold = TRUE) + xl_fill(background = "yellow")').
width_pixels, height_pixels	The same geometry given in pixels instead. Give one or the other, not both. Excel stores character units and points, so the pixel value is converted on the way in — read back, a width set as 100 pixels is 13.57 character units.

collapsed	Logical; draw this column/row as the collapsed summary of the group beside it. Excel does not derive this — the rows or columns of the group itself need 'hidden = TRUE' as well, exactly as clicking the grouping symbol would leave them.
na	What to write in this column where a value has none, overriding the workbook's [xl_properties()]'(na =)'. 'NA' (the default) inherits it. Columns are where this usually belongs: a substitute that suits a numeric column rarely suits a date one.
rows	Rows to target: a numeric vector of 1-based data-row indices (row 1 is the first data row, ignoring the header).
height	Row height (in points).

Value

An 'xl_col_spec' / 'xl_row_spec' object (also an [xl_format]).

See Also

[xl_sheet], [xl_format]

Other worksheet layout: `xl_outline()`, `xl_page_setup()`, `xl_sheet()`, `xl_sheet_view()`

Examples

```
xl_col_spec("revenue", width = 14, format = xl_num_format("#,##0.00"))
xl_col_spec(c(1, 2), width = 10) + xl_font(bold = TRUE)
xl_col_spec("logo", width_pixels = 100)
xl_row_spec(1, height = 24, format = xl_font(bold = TRUE))
xl_row_spec(1, height_pixels = 40)
```

---

xl_comment *Create a cell comment*

Description

'xl_comment()' builds a comment (note) that can be attached to a cell via [xl_cell_general()]'s 'comment' argument. The simplest form is just text ('xl_cell_general(value = x, comment = "see note")'); 'xl_comment()' adds options such as the author, initial visibility, box size and position, and styling.

Excel comment boxes support only a **background color** and a **font name/size/family**. These are supplied by reusing the formatting engine via the 'format' argument (e.g. 'xl_font(name = "Arial", size = 10) + xl_fill(background = "lightyellow")'); any other format property (bold, borders, number formats, ...) is not supported by comments and triggers a warning.

Usage

```
xl_comment(
  value,
  format = NULL,
  author = NA,
  visible = NA,
  width_pixels = NA,
  height_pixels = NA,
  x_scale = NA,
  y_scale = NA,
  start_row = NA,
  start_col = NA,
  x_offset = NA,
  y_offset = NA
)
```

Arguments

value	A single string: the comment's text. Anything with an [as.character()] method is accepted, so a cell built for a sheet can be reused here.
format	An optional [xl_format]; only its fill background color and font name/size/family are used (see [xl_fill()], [xl_font()]). Other properties are unsupported and warned about.
author	Comment author (shown in Excel's status bar). Defaults to the sheet/workbook 'comment_author' when unset (see [xl_sheet()], [xl_properties()]).
visible	Initial visibility: 'NA' follows the sheet default (comments are hidden unless 'show_comments' is set), 'TRUE' shows this comment, 'FALSE' hides it.
width_pixels, height_pixels	Comment box size in pixels (defaults 128 x 74).
x_scale, y_scale	Box scale factors.
start_row, start_col	Zero-based anchor cell of the box (by default a comment sits one row up and one column right of its cell).
x_offset, y_offset	Pixel offset of the box from its anchor.

Value

An 'xl_comment' object.

See Also

[xl_cell_general], [xl_format]

Other cell content: is_xl_comment(), xl_cell_general(), xl_formula(), xl_rich_run(), xl_rich_string()

Examples

```
# plain text
xl_comment("Double-check this figure")

# with author and styling reused from the format engine
xl_comment("Estimate", author = "Finance",
           format = xl_font(name = "Arial", size = 10) +
                    xl_fill(background = "lightyellow"))
```

`xl_conditional` *Format cells according to their contents*

Description

Excel's conditional formatting, in four flavours:

* 'xl_cond_cell()' — a rule with a format: comparisons, text matches, time periods, above/below average, top/bottom N, duplicates, blanks, errors, or an arbitrary formula. * 'xl_cond_scale()' — a two- or three-colour scale across the range. * 'xl_cond_bar()' — in-cell data bars. * 'xl_cond_icons()' — one of Excel's built-in icon sets.

Pass one or a list of them as 'xl_sheet(conditional =)'.

Usage

```
xl_cond_cell(
  range,
  type = NA,
  criteria = NA,
  value = NULL,
  min = NULL,
  max = NULL,
  format = NULL,
  stop_if_true = NA,
  multi_range = NA
)

xl_cond_scale(
  range,
  colors = c("red", "yellow", "green"),
  values = NULL,
  rule_types = NULL,
  stop_if_true = NA,
  multi_range = NA
)

xl_cond_bar(
  range,
```

```
  color = NA,
  values = NULL,
  rule_types = NULL,
  solid = NA,
  negative_color = NA,
  border_color = NA,
  negative_border_color = NA,
  no_border = NA,
  direction = NA,
  axis = NA,
  axis_color = NA,
  bar_only = NA,
  stop_if_true = NA,
  multi_range = NA
)

xl_cond_icons(
  range,
  style = "3_traffic_lights",
  reverse = NA,
  icons_only = NA,
  stop_if_true = NA,
  multi_range = NA
)
```

Arguments

range	The cells the rule applies to: an Excel range string such as ‘"B2:B100"‘, a single cell, or a ‘list(rows = , cols =)‘ spec.
type	The kind of rule, when it cannot be inferred from ‘criteria‘: ‘"cell"‘, ‘"text"‘, ‘"time_period"‘, ‘"average"‘, ‘"top"‘, ‘"bottom"‘, ‘"duplicate"‘, ‘"unique"‘, ‘"blanks"‘, ‘"no_blanks"‘, ‘"errors"‘, ‘"no_errors"‘ or ‘"formula"‘.
criteria	How the rule decides, which depends on ‘type‘: ‘"=="‘, ‘"!="‘, ‘">"‘, ‘"<"‘, ‘">="‘, ‘"<="‘, ‘"between"‘, ‘"not between"‘ for ‘"cell"‘; ‘"contains"‘, ‘"not contains"‘, ‘"begins with"‘, ‘"ends with"‘ for ‘"text"‘; ‘"yesterday"‘, ‘"today"‘, ‘"tomorrow"‘, ‘"last 7 days"‘, ‘"last week"‘, ‘"this week"‘, ‘"next week"‘, ‘"last month"‘, ‘"this month"‘, ‘"next month"‘ for ‘"time_period"‘; ‘"above"‘, ‘"below"‘, ‘"above or equal"‘, ‘"below or equal"‘ and the ‘"N std dev above"‘ / ‘"below"‘ variants for ‘"average"‘; and ‘"percent"‘ for ‘"top"‘ / ‘"bottom"‘. Pairing a criteria with the wrong ‘type‘ is an error — Excel would accept the file and silently ignore the rule.
value	What the criteria compares against: a number, a string (for the text criteria), or an ‘"=..."‘ formula. For ‘"top"‘ / ‘"bottom"‘ it is the N. Use ‘min‘ and ‘max‘ for ‘"between"‘ / ‘"not between"‘.
min, max	The two bounds for ‘"between"‘ / ‘"not between"‘.
format	The [xl_format] applied to cells that match.
stop_if_true	Logical; if this rule matches, skip the later rules on the same cells.

`multi_range`	A further set of ranges the rule also covers, as an Excel multi-range string such as ‘"B3:K6 B9:K12"‘.
`colors`	Two or three colours for the scale, from lowest to highest. Two gives a two-colour scale, three a three-colour scale.
`values`	Optional values marking where each colour sits, in the same order as ‘colors‘. Defaults to the range’s minimum, midpoint and maximum.
`rule_types`	How each entry of ‘values‘ is interpreted: ‘"minimum"‘, ‘"maximum"‘, ‘"number"‘, ‘"percent"‘, ‘"percentile"‘, ‘"formula"‘, ‘"auto_min"‘ or ‘"auto_max"‘.
`color`	The bar’s fill colour.
`solid`	Logical; a solid bar rather than Excel’s default gradient.
`negative_color`, `border_color`, `negative_border_color`, `axis_color`	Colours for the negative portion, the bar border, the negative border, and the axis line.
`no_border`	Logical; draw the bar without a border.
`direction`	‘"context"‘ (follow the sheet), ‘"left to right"‘ or ‘"right to left"‘.
`axis`	Where the zero axis sits: ‘"automatic"‘, ‘"midpoint"‘ or ‘"none"‘.
`bar_only`	Logical; show the bar without the cell’s value.
`style`	Which built-in icon set: one of ‘"3_arrows"‘, ‘"3_arrows_gray"‘, ‘"3_flags"‘, ‘"3_traffic_lights"‘, ‘"3_traffic_lights_rimmed"‘, ‘"3_signs"‘, ‘"3_symbols_circled"‘, ‘"3_symbols"‘, ‘"4_arrows"‘, ‘"4_arrows_gray"‘, ‘"4_red_to_black"‘, ‘"4_ratings"‘, ‘"4_traffic_lights"‘, ‘"5_arrows"‘, ‘"5_arrows_gray"‘, ‘"5_ratings"‘ or ‘"5_quarters"‘. These are Excel’s own icons, not images, so nothing is embedded in the file.
`reverse`	Logical; reverse the order the icons are assigned in.
`icons_only`	Logical; show the icon without the cell’s value.

Value

An ‘xl_conditional‘ object.

See Also

[xl_sheet], [xl_format]

Other worksheet features: `xl_filter()`, `xl_filter_keep()`, `xl_merge()`, `xl_table()`, `xl_table_column()`, `xl_validation()`

Examples

```
xl_cond_cell("B2:B100", criteria = ">", value = 100, format = xl_fill(background = "red"))
xl_cond_cell("C2:C100", type = "text", criteria = "contains", value = "urgent",
             format = xl_font(bold = TRUE))
xl_cond_cell("D2:D100", type = "duplicate",
             format = xl_fill(background = "yellow"))
xl_cond_scale("C2:C100", colors = c("red", "yellow", "green"))
xl_cond_scale("C2:C100", colors = c("white", "steelblue"))
xl_cond_bar("D2:D100", color = "steelblue")
```

```
xl_cond_bar("D2:D100", color = "green", solid = TRUE, bar_only = TRUE)
xl_cond_icons("E2:E100", style = "3_traffic_lights")
xl_cond_icons("E2:E100", style = "5_ratings", icons_only = TRUE)
```

`xl_filter` *Filter an autofilter column, hiding the rows that do not match*

Description

‘xl_filter()‘ sets the criteria on one autofilter column *and* hides the rows that do not match. Both halves are necessary: Excel stores the criteria and the hidden rows separately and does not apply a filter when a file is opened, so criteria alone produce a sheet that looks filtered but shows every row.

Because writexl decides which rows to hide, it reproduces Excel’s own matching rules, which were measured in Excel rather than assumed. Those rules depend on which of two forms the filter takes:

* ‘"=="‘ (with no wildcard) and ‘"blanks"‘, and any ‘list‘, are written as a **value list**. Excel matches these against the text a cell *displays*, case-insensitively — so ‘"=="‘ with ‘10‘, with ‘"10"‘, or a ‘list‘ of ‘"10"‘ all match both the number ‘10‘ and the string ‘"10"‘. * every other criteria, including ‘"=="‘ with a ‘*‘ or ‘¿ in it, is written as a **typed comparison**. If the value is a number the comparison is numeric and a text cell never satisfies it (except ‘"!="‘, which a text cell satisfies because it is not that number). If the value is text the comparison is textual, case-insensitive, with ‘*‘ and ‘¿ as wildcards, and a number cell never satisfies it.

The consequence worth knowing is that ‘"=="‘ with ‘"10"‘ keeps the number ‘10‘, while ‘"=="‘ with ‘"1*"‘ keeps nothing on a numeric column: the wildcard changes the form, and so the rule.

Blank means an empty cell or an empty string; ‘"non-blanks"‘ is its exact complement. Blanks are excluded by every comparison except ‘"!="‘, which keeps them — a blank is not equal to anything. A mixed-type column built with [xl_cell_general()] is matched cell by cell, each by its own type.

Usage

```
xl_filter(
  col,
  criteria = NA,
  value = NULL,
  criteria2 = NA,
  value2 = NULL,
  and_or = "and",
  list = NULL
)
```

Arguments

`col`	The column to filter: a name or a 1-based position.
`criteria`	One of ‘"=="‘, ‘"!="‘, ‘">"‘, ‘"<"‘, ‘">="‘, ‘"<="‘, ‘"blanks"‘ or ‘"non-blanks"‘. The four magnitude comparisons need a numeric value; writexl will not guess how Excel orders text.

`value` The value to compare against. In a text comparison '*' matches any run of characters and '¿ any single one.

`criteria2, value2`
An optional second rule for the same column.

`and_or` How the two rules combine: '"and"' (default) or '"or"'.

`list` Instead of a criteria, keep only rows whose value is in this character vector. Matched case-insensitively, as Excel does.

Value

An 'xl_filter' object.

Limitations

A value list matches displayed text, which writexl can only predict for the General format. Filtering a 'Date' or 'POSIXct' column by '"=="' or 'list' is therefore refused — use a comparison such as '">="'. For the same reason a numeric column carrying a custom number format (say two decimal places, or a currency symbol) may display differently from what writexl compares, so prefer a comparison there too. Columns whose cells hold formulas cannot be filtered at all, since writexl does not know what Excel would compute.

See Also

[xl_sheet], [xl_filter_keep]

Other worksheet features: `xl_conditional`, `xl_filter_keep()`, `xl_merge()`, `xl_table()`, `xl_table_column()`, `xl_validation()`

Examples

```
xl_filter("qty", ">", 100)
xl_filter("qty", ">", 100, "<", 200)          # between, via two rules
xl_filter("fruit", "==", "ap*")               # wildcard: typed comparison
xl_filter("fruit", list = c("apple", "banana"))
xl_filter("qty", "==", "10")                  # value list: matches the
                                              # number 10 and the text "10"
```

---

xl_filter_keep *Which rows an Excel autofilter would leave visible*

Description

'xl_filter_keep()' answers, for a data frame and a set of [xl_filter()] rules, the question 'xl_sheet(filter =)' has to answer internally: which rows does Excel leave visible? It writes nothing — it is the matching rule on its own, exported because reproducing Excel's filter semantics is hard to get right and useful outside writing a file.

The rules are described in detail under [xl_filter()], and were established by measurement rather than from documentation: a workbook was written with criteria set and no rows hidden, opened in Excel, and Data > Reapply pressed so that Excel computed each match itself.

Usage

```
xl_filter_keep(data, filter)
```

Arguments

data	A data frame whose columns the filters name.
filter	One [xl_filter()], or a list of them. Filters on different columns combine with AND, as they do in Excel.

Value

A logical vector with one element per row of ‘data‘, ‘TRUE‘ where the row stays visible.

Accuracy

This function tracks Excel’s *observed* behaviour, so a case found to disagree with Excel is treated as a bug and fixed, which may change the rows it returns. One limitation is known: a filter written as a value list (‘"=="‘ without a wildcard, or ‘list‘) matches the text a cell **displays**, which writexl can only predict for the General format. A numeric column carrying a custom number format may therefore display differently from what is compared here — prefer a comparison such as ‘">="‘ on such a column. Dates are refused outright for the same reason. See [xl_filter()].

See Also

[xl_filter], [xl_sheet]

Other worksheet features: xl_conditional, xl_filter(), xl_merge(), xl_table(), xl_table_column(), xl_validation()

Examples

```
sales <- data.frame(fruit = c("apple", "banana", "cherry"),
                    qty = c(5, 150, 300))
xl_filter_keep(sales, xl_filter("qty", ">", 100))
sales[xl_filter_keep(sales, xl_filter("fruit", "==", "*a*")), ]

# filters on different columns combine with AND
xl_filter_keep(sales, list(xl_filter("qty", ">", 100),
                           xl_filter("fruit", "==", "b*")))
```

---

xl_format *Combine cell-formatting groups into a single format*

Description

'xl_format()' merges any number of [xl_format] objects (typically the single-group objects returned by [xl_font], [xl_fill], [xl_border], [xl_align], [xl_num_format] and [xl_protection]) into one combined format. The '+' operator does the same for two formats.

Merging is right-biased and works property-by-property: where two formats set the *same* property the later one wins, but properties set by only one side are all preserved. So partial groups accumulate rather than overwrite.

Usage

```
xl_format(..., quote_prefix = NA, hyperlink = NA)

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

Arguments

`...`	[xl_format] objects (and/or 'NULL's, which are ignored). May also include the scalar flags 'quote_prefix' and 'hyperlink'.
`quote_prefix`	Logical; treat the cell contents as literal text (as if prefixed with a single quote in Excel).
`hyperlink`	Logical; apply the internal hyperlink style flag (advanced).
`e1, e2`	[xl_format] objects to combine.

Value

A combined [xl_format] object.

See Also

[xl_font], [xl_fill], [xl_border], [xl_align], [xl_num_format], [xl_protection], [xl_color]

Other cell formatting: `is_xl_format()`, `xl_color()`, `xl_format_groups`

Examples

```
xl_format(xl_font(bold = TRUE), xl_border(bottom = "thin"),
          xl_num_format("#,##0.00"))

# equivalent, using +
xl_font(bold = TRUE) + xl_border(bottom = "thin") + xl_num_format("#,##0.00")
```

xl_format_groups *Cell formatting groups*

Description

These constructors build [xl_format] objects, each populating one group of Excel cell-formatting properties. Every property defaults to ‘NA‘, meaning "leave unset"; unset properties are simply not written. Enum-like arguments take lowercase strings and are validated against a fixed set of choices.

Each constructor returns a full ‘xl_format‘, so a single group can be used on its own (‘xl_cell_general(1, format = xl_font(bold = TRUE))‘), and groups can be combined with ‘+‘ (see [xl_format]).

Usage

```
xl_font(
  bold = NA,
  italic = NA,
  color = NA,
  size = NA,
  name = NA,
  underline = NA,
  strikeout = NA,
  script = NA,
  family = NA,
  charset = NA,
  outline = NA,
  shadow = NA,
  condense = NA,
  extend = NA,
  scheme = NA,
  theme = NA,
  color_indexed = NA,
  font_only = NA
)

xl_fill(background = NA, foreground = NA, pattern = NA, transparency = NA)

xl_border(
  all = NA,
  left = NA,
  right = NA,
  top = NA,
  bottom = NA,
  color = NA,
  left_color = NA,
  right_color = NA,
  top_color = NA,
```

```
  bottom_color = NA,
  diagonal = NA,
  diagonal_style = NA,
  diagonal_color = NA,
  transparency = NA
)

xl_align(
  horizontal = NA,
  vertical = NA,
  wrap = NA,
  rotation = NA,
  indent = NA,
  shrink = NA,
  reading_order = NA
)

xl_num_format(format = NA, index = NA)

xl_protection(locked = NA, hidden = NA)
```

Arguments

`bold`, `italic`, `strikeout`, `outline`, `shadow`, `condense`, `extend`, `font_only`	Logical font flags.
`color`, `background`, `foreground`, `left_color`, `right_color`, `top_color`, `bottom_color`, `diagonal_color`	A color: an R color name, a hex string, or an integer (see [xl_color]).
`size`	Font size in points (1–409).
`name`	Font name, e.g. ‘"Calibri"‘.
`underline`	One of ‘"none"‘, ‘"single"‘, ‘"double"‘, ‘"single-accounting"‘, ‘"double-accounting"‘.
`script`	One of ‘"super"‘, ‘"sub"‘.
`family`, `charset`, `theme`, `color_indexed`	Advanced integer font properties (rarely needed).
`scheme`	Font scheme string (advanced).
`pattern`	Fill pattern, e.g. ‘"solid"‘, ‘"light-gray"‘ (18 choices). When ‘background‘ is supplied and ‘pattern‘ is unset, a ‘"solid"‘ pattern is assumed.
`transparency`	Percentage transparency, 0–100. **Charts only**: Excel has no transparency for a cell’s fill or border, so this is ignored everywhere except a chart’s line and fill (see [xl_chart()]).
`all`	Border style applied to all four sides at once (one of ‘"none"‘, ‘"thin"‘, ‘"medium"‘, ‘"dashed"‘, ‘"dotted"‘, ‘"thick"‘, ‘"double"‘, ‘"hair"‘, ‘"medium-dashed"‘, ‘"dash-dot"‘, ‘"medium-dash-dot"‘, ‘"dash-dot-dot"‘, ‘"medium-dash-dot-dot"‘, ‘"slant-dash-dot"‘).
`left`, `right`, `top`, `bottom`	Per-side border styles (override ‘all‘).

diagonal	Diagonal border direction: ‘"up"‘, ‘"down"‘, ‘"up-down"‘.
diagonal_style	Diagonal border style (same choices as ‘all‘).
horizontal	Horizontal alignment: ‘"left"‘, ‘"center"‘, ‘"right"‘, ‘"fill"‘, ‘"justify"‘, ‘"center-across"‘, ‘"distributed"‘.
vertical	Vertical alignment: ‘"top"‘, ‘"bottom"‘, ‘"center"‘, ‘"justify"‘, ‘"distributed"‘.
wrap	Logical; wrap text in the cell.
rotation	Text rotation in degrees (-90..90, or 270).
indent	Integer indentation level.
shrink	Logical; shrink text to fit.
reading_order	One of ‘"default"‘, ‘"ltr"‘, ‘"rtl"‘.
format	A number-format string, e.g. ‘"#,##0.00"‘, ‘"0 ‘"yyyy-mm-dd"‘.
index	An Excel built-in number-format index (alternative to ‘format‘).
locked	Logical; whether the cell is locked (Excel’s default is ‘TRUE‘). Only takes effect when the worksheet is protected. ‘FALSE‘ unlocks the cell.
hidden	Logical; hide the cell’s formula.

Value

An [xl_format] object.

See Also

[xl_format], [xl_color]

Other cell formatting: `is_xl_format()`, `xl_color()`, `xl_format()`

Examples

```
xl_font(bold = TRUE, color = "navy", size = 12)
xl_fill(background = "#FFF2CC")
xl_border(all = "thin", color = "gray")
xl_align(horizontal = "center", vertical = "top", wrap = TRUE)
xl_num_format("#,##0.00")
xl_protection(locked = FALSE)
```

`xl_formula` *Excel Types*

Description

* ‘xl_formula(x)‘ — wraps a character vector of Excel formulas (each must start with ‘"="‘). The formulas are written to the xlsx file as-is and are recalculated by Excel on open.

* ‘xl_hyperlink(url, name)‘ — convenience wrapper that builds an Excel ‘=HYPERLINK(url, name)‘ **formula** for each element. Because the hyperlink is stored as a formula, it is readable by [readxl::read_xlsx()], which returns the formula text. Display text is controlled by the ‘name‘ argument.

* ‘xl_hyperlink_cell(url, value)‘ — creates a **native cell-level hyperlink** using ‘worksheet_write_url_opt()‘ from libxlsxwriter. The URL is stored as metadata attached to the cell, not in the formula bar. An optional ‘value‘ argument provides the display text shown in the cell. A tooltip and further options can be set by passing a named list to ‘xl_cell_general()‘ directly. **Note:** [readxl::read_xlsx()] cannot read cell-level hyperlinks and returns ‘NA‘ for those cells. Use ‘xl_hyperlink()‘ instead when round-tripping through readxl is required.

Usage

```
xl_formula(x, format = NULL)

xl_hyperlink(url, value = NULL, format = NULL, name = NULL)

xl_hyperlink_cell(url, value = NULL, format = NULL)
```

Arguments

x	character vector to be interpreted as formula
format	An optional [xl_format] (or list of ‘xl_format‘, one per element) applied to the cells. See [xl_format].
url	character vector of URLs. Use ‘NA‘ to produce a blank cell.
value	character vector (or ‘NULL‘) of display text shown in the cell instead of the URL. When ‘NULL‘ the URL itself is shown. Recycled to the length of ‘url‘, and automatically ‘NA‘ for cells whose URL is ‘NA‘. The same argument name is used by [xl_hyperlink_cell()] and [xl_cell_general()].
name	**Deprecated.** The former spelling of ‘value‘, kept for backward compatibility. Supplying it warns and points at ‘value‘; supplying both is an error, since they mean the same thing. ‘value‘ has taken the argument position ‘name‘ used to occupy, so code that passed the display text positionally keeps working unchanged.

Details

Create special column types to write to a spreadsheet.

See Also

Other cell content: `is_xl_comment()`, `xl_cell_general()`, `xl_comment()`, `xl_rich_run()`, `xl_rich_string()`

Examples

```
df <- data.frame(
  name = c("UCLA", "Berkeley", "Jeroen"),
  founded = c(1919, 1868, 2030),
 website = xl_hyperlink(c("http://www.ucla.edu", "http://www.berkeley.edu", NA), "homepage")
)
df$age <- xl_formula('=(YEAR(TODAY()) - INDIRECT("B" & ROW()))')
write_xlsx(df, 'universities.xlsx')

# xl_hyperlink_cell() stores the URL as native cell metadata.
# readxl cannot read these cells, but they display cleanly in Excel.
df2 <- data.frame(
  name = c("UCLA", "Berkeley"),
  website = xl_hyperlink_cell(c("http://www.ucla.edu", "http://www.berkeley.edu"),
                              value = "homepage")
)
write_xlsx(df2, 'universities2.xlsx')

# cleanup
unlink(c('universities.xlsx', 'universities2.xlsx'))
```

---

xl_image *Insert an image into a worksheet*

Description

'xl_image()' places an image on a sheet, either floating over the cells and anchored to one of them (the default) or, with 'embed = TRUE', inside a cell. Pass one or a list of them as 'xl_sheet(image =)'.

The image may be a file path or a raw vector, which is convenient when a plot has just been written by a graphics device and never touched the disk. PNG, JPEG, GIF and BMP are supported — the formats Excel reads — and the format is detected from the file's own bytes rather than its extension.

Usage

```
xl_image(
  image,
  at = "A1",
  scale = 1,
  offset = NULL,
  position = "move_and_size",
  description = NULL,
  decorative = FALSE,
  url = NULL,
  tip = NULL,
  embed = FALSE,
  format = NULL
)
```

Arguments

image	The image, in any of four shapes: a path to a PNG, JPEG, GIF or BMP; a raw vector holding one of those encoded; a ‘raster‘ (or anything [grDevices::as.raster()] accepts, such as a colour matrix or an RGB/RGBA array); or a ‘nativeRaster‘. The last two are what [graphics::rasterImage()] draws, so anything you can plot can be written.
at	The cell the image is anchored to, such as ‘"B2"‘, or a ‘list(rows = , cols =)‘ spec selecting a single cell.
scale	Scale factor: one number for both axes, or ‘c(x, y)‘.
offset	Offset from the anchor cell’s top-left corner in pixels, as ‘c(x, y)‘.
position	How the image behaves when rows and columns change size: ‘"move_and_size"‘ (the default), ‘"move_dont_size"‘, ‘"dont_move_dont_size"‘, ‘"move_and_size_after"‘, or ‘"default"‘ for Excel’s own default. Ignored when ‘embed = TRUE‘.
description	Alt text, for screen readers. Excel defaults it to the file name; ‘""‘ writes none.
decorative	Mark the image as decorative, so screen readers skip it. Excel does not write a description for a decorative image.
url	An optional hyperlink the image links to.
tip	An optional mouseover tip for ‘url‘.
embed	Place the image inside the cell rather than floating above it. Requires a version of Excel that supports images in cells.
format	An [xl_format] for the cell holding an embedded image. Only meaningful with ‘embed = TRUE‘.

Value

An ‘xl_image‘ object.

Floating versus embedded

An inserted image floats above the grid: it has a position but occupies no cell, and ‘position‘ decides whether it moves and resizes as rows and columns change. An embedded image (‘embed = TRUE‘) lives *in* a cell and sizes with it, which is the Excel 365 "place in cell" behaviour. Excel versions without that feature show ‘#VALUE¡ in place of an embedded image, so it is worth choosing deliberately.

Caveats inherited from Excel

An image’s scaling can shift if it crosses a row whose height changed — for a taller font or wrapped text — so set the height explicitly with [xl_row_spec()] for rows an image spans. BMP is supported only for backward compatibility and must be 24-bit true colour; prefer PNG. SVG is refused, because Excel stores it converted to PNG anyway.

See Also

[xl_sheet]

Other images and charts: xl_chart(), xl_chart_axis(), xl_chart_error_bars(), xl_chart_labels(), xl_chart_legend(), xl_chart_marker(), xl_chart_series(), xl_chart_table(), xl_chart_trendline(), xl_chartsheet()

Examples

```
logo <- system.file("help", "figures", "logo.png", package = "writexl")
if (nzchar(logo)) {
  xl_image(logo, at = "C2", scale = 0.5)
  xl_image(logo, at = "C2", url = "https://example.com", tip = "Home")
}
```

---

xl_merge *Merge a range of cells*

Description

‘xl_merge()‘ merges a rectangle of cells into one, as Excel’s "Merge and Centre" does. Pass one or a list of them as ‘xl_sheet(merge =)‘.

A merged range holds a single value, so ‘xl_merge()‘ carries its own ‘value‘ rather than taking it from the data frame. Merging over cells the data frame filled keeps only the merged value, exactly as merging in Excel discards everything but the top-left value.

Usage

```
xl_merge(range, value = NULL, format = NULL)
```

Arguments

range	The cells to merge: an Excel range string such as ‘"A1:C1"‘, or a ‘list(rows = , cols =)‘ spec. It must cover more than one cell — Excel has no single-cell merge.
value	The value shown in the merged cell: a string, or anything with an [as.character()] method such as an [xl_rich_string()]. ‘NULL‘ leaves the cell empty.
format	An optional [xl_format] applied to the whole merged range. Merged cells usually want ‘xl_align(horizontal = "center")‘.

Value

An ‘xl_merge‘ object.

See Also

[xl_sheet], [xl_format]

Other worksheet features: xl_conditional, xl_filter(), xl_filter_keep(), xl_table(), xl_table_column(), xl_validation()

Examples

```
xl_merge("A1:C1", "Quarterly results",
         format = xl_align(horizontal = "center") + xl_font(bold = TRUE))

df <- data.frame(a = 1:3, b = 4:6)
sheet <- xl_sheet(df, merge = xl_merge("A5:B5", "Total",
                                       format = xl_font(bold = TRUE)))
tmp <- write_xlsx(list(Data = sheet))
```

---

xl_outline *Control how outline (grouping) symbols are drawn*

Description

Grouping itself comes from ‘level‘ in [xl_col_spec()] / [xl_row_spec()]. ‘xl_outline()‘ only changes how the controls are *displayed*, which is rarely needed — the defaults match Excel’s own.

Usage

```
xl_outline(
  visible = TRUE,
  symbols_below = TRUE,
  symbols_right = TRUE,
  auto_style = FALSE
)
```

Arguments

visible	Show the outline symbols at all. ‘FALSE‘ keeps the grouping (and so the collapsing) but hides the +/- controls.
symbols_below	Put the summary row *below* the detail rows, which is Excel’s default. ‘FALSE‘ puts it above.
symbols_right	Put the summary column to the *right* of the detail columns, Excel’s default. ‘FALSE‘ puts it to the left.
auto_style	Apply Excel’s automatic outline styling to the grouped rows and columns.

Value

An ‘xl_outline‘ object, for [xl_sheet()]’s ‘outline‘ argument.

See Also

[xl_sheet], [xl_colrow_spec]

Other worksheet layout: xl_colrow_spec, xl_page_setup(), xl_sheet(), xl_sheet_view()

Examples

```
xl_outline(symbols_below = FALSE)
xl_outline(visible = FALSE)
```

`xl_page_setup` *How a worksheet prints*

Description

'xl_page_setup()' collects Excel's page-layout settings for one worksheet: orientation, paper size, margins, scaling, centring, the print options, and the header and footer. Pass it as 'xl_sheet(page =)'.

None of these affect the cell data — they change only how the sheet prints and how it looks in Excel's page-break preview.

Usage

```
xl_page_setup(
  orientation = NA,
  paper = NA,
  margins = NULL,
  scale = NA,
  fit_to = NULL,
  center_horizontally = NA,
  center_vertically = NA,
  header = NA,
  footer = NA,
  header_margin = NA,
  footer_margin = NA,
  header_image = NULL,
  footer_image = NULL,
  page_view = NA,
  first_page = NA,
  across = NA,
  black_and_white = NA,
  row_col_headers = NA,
  print_area = NULL,
  repeat_rows = NA,
  repeat_cols = NA,
  h_breaks = NULL,
  v_breaks = NULL
)
```

Arguments

`orientation`	‘"portrait"‘ or ‘"landscape"‘.
`paper`	Paper size: a name (‘"A4"‘, ‘"letter"‘, ‘"legal"‘, ‘"tabloid"‘, ‘"ledger"‘, ‘"statement"‘, ‘"executive"‘, ‘"A3"‘, ‘"A5"‘, ‘"B4"‘, ‘"B5"‘, ‘"folio"‘, ‘"quarto"‘, ‘"default"‘), or an Excel paper-type integer for the envelope and specialist sizes that have no name here.
`margins`	Page margins in inches: a single number for all four sides, four numbers in the order left, right, top, bottom, or a named vector using any of ‘left‘, ‘right‘, ‘top‘, ‘bottom‘.
`scale`	Print scaling as a percentage (10–400). Ignored by Excel when ‘fit_to‘ is set.
`fit_to`	Fit the printout to ‘c(width, height)‘ pages. A ‘0‘ means "as many pages as needed in that direction", so ‘c(width = 1, height = 0)‘ is Excel’s "fit all columns on one page".
`center_horizontally, center_vertically`	Logical; centre the printed output on the page.
`header, footer`	Header and footer text, at most 255 characters, using Excel’s own codes: ‘&L‘, ‘&C‘, ‘&R‘ start the left, centre and right sections, ‘&P‘ is the page number, ‘&N‘ the page count, ‘&D‘ the date, ‘&A‘ the sheet name, and ‘&&‘ a literal ampersand. For example ‘"&LQ1 report&RPage &P of &N"‘. Image placeholders (‘&G‘ / ‘&[Picture]‘) are not supported yet and are rejected.
`header_margin, footer_margin`	Header/footer margin in inches (Excel’s default is 0.3). Must be greater than 0.
`header_image, footer_image`	Images to place in the header or footer, named by position: ‘list(left = , center = , right =)‘. Each may be a file path, a raw vector, or an in-memory image, exactly as [xl_image()] accepts. Each image needs a matching ‘&G‘ placeholder in the corresponding section of ‘header‘/‘footer‘ — ‘"&L&G"‘ puts one on the left — and the counts must agree.
`page_view`	Logical; open the sheet in Excel’s page-layout view rather than normal view.
`first_page`	The page number to start numbering from.
`across`	Logical; print pages left-to-right before top-to-bottom (Excel’s "over, then down").
`black_and_white`	Logical; print without colour.
`row_col_headers`	Logical; print the row numbers and column letters.
`print_area`	The range to print: an Excel range string such as ‘"A1:F50"‘, or a ‘list(rows = , cols =)‘ spec naming data rows and columns, as elsewhere in writexl.
`repeat_rows, repeat_cols`	Rows/columns to repeat at the top or left of every printed page. Either a count (‘repeat_rows = 1‘ repeats the first sheet row, which is the header when ‘col_names = TRUE‘) or a range string (‘"1:2"‘, ‘"A:B"‘). Note these count *sheet* rows from 1 with the header included, unlike [xl_row_spec()], which indexes data rows — repeating the header row is the usual reason to use this, and data-row numbering could not name it.

h_breaks, v_breaks
: Manual page breaks: 1-based sheet positions at which a new page starts, so ‘h_breaks = 21‘ breaks between rows 20 and 21. ‘v_breaks‘ also accepts column letters. Positions must be 2 or greater, and at most 1023 breaks are allowed. Excel ignores manual breaks when ‘fit_to‘ is set, which warns.

Value

An ‘xl_page_setup‘ object.

See Also

[xl_sheet], [write_xlsx]

Other worksheet layout: xl_colrow_spec, xl_outline(), xl_sheet(), xl_sheet_view()

Examples

```
xl_page_setup(orientation = "landscape", paper = "A4",
              fit_to = c(width = 1, height = 0))

# margins in inches, and a header with page numbers
xl_page_setup(margins = c(left = 1, right = 1),
              header = "&LQuarterly report&RPage &P of &N")

df <- data.frame(x = 1:3)
tmp <- write_xlsx(list(Data = xl_sheet(df, page = xl_page_setup(
  orientation = "landscape", header = "&CDraft"
))))
```

---

xl_properties *Workbook properties, defaults, and metadata*

Description

‘xl_properties()‘ collects everything that applies at the *workbook* level: document metadata, a few native workbook settings, and the formatting defaults that used to be hard-coded. The formatting defaults are ordinary [xl_format] objects, so you can override them (e.g. change the header style or the default date format) simply by passing a different ‘xl_format‘.

The ‘default_format‘ is cascaded *under* every cell (an emulated workbook-wide default: libxlsxwriter has no native "Normal style" setter, so it is merged beneath each cell/column format). ‘header_format‘ styles the header row and ‘hyperlink_format‘ styles cell hyperlinks; both are also cascaded over ‘default_format‘.

Usage

```
xl_properties(
  title = NA,
  subject = NA,
  author = NA,
  manager = NA,
  company = NA,
  category = NA,
  keywords = NA,
  comments = NA,
  status = NA,
  hyperlink_base = NA,
  created = NA,
  na = NA,
  custom = NULL,
  read_only = FALSE,
  window_size = NULL,
  names = NULL,
  default_format = xl_format(),
  header_format = xl_font(bold = TRUE) + xl_align(horizontal = "center"),
  hyperlink_format = xl_font(color = "blue", underline = "single"),
  date_format = xl_num_format("yyyy-mm-dd"),
  datetime_format = .default_datetime_format(),
  date_col_width = 20,
  datetime_col_width = 20,
  header_row_height = 15
)
```

Arguments

`title`, `subject`, `author`, `manager`, `company`, `category`, `keywords`, `comments`, `status`, `hyperlink_base`
: Document metadata strings (Excel's "Properties" dialog).

`created`
: The workbook's creation timestamp, a 'Date' or 'POSIXct'. Left 'NA' libxlsxwriter stamps the moment the file is written, which is what makes two runs over the same data differ; pinning it makes them byte-identical.

`na`
: What to write where a value has none — an 'NA' or 'NaN' in the data, or a cell with nothing in it at all. 'NA' (the default) leaves the cell blank, which is what writexl has always done. Anything else is written in its place, keeping its own type: 'na = "Not available"' writes a string, 'na = 0' a number. A column or an individual cell can override it — see [xl_col_spec()] and [xl_cell_general()]. Note that a non-blank 'na' in a numeric column makes that column mixed, so a reader such as [readxl::read_xlsx()] returns the whole column as character.

`custom`
: A named list of custom document properties. Values may be character, integer, numeric, logical, 'Date' or 'POSIXct'. A 'Date' or 'POSIXct' is written as a real datetime property (not as text) and follows the same workbook-wide time zone rule as datetime cells, described below.

`read_only` Logical; mark the workbook read-only recommended.

`window_size` Optional integer vector ‘c(width, height)‘ for the workbook window size.

`names` A named list of workbook-scoped defined names, each a formula string (e.g. ‘list(tax = "=0.2")‘).

`default_format` An [xl_format] cascaded under every cell (default: none).

`header_format` An [xl_format] for the header row (default: bold, centered).

`hyperlink_format`
An [xl_format] for cell hyperlinks (default: blue, underlined), or ‘NULL‘ for no hyperlink styling at all. ‘NULL‘ is the only way to write an unstyled hyperlink: an empty ‘xl_format()‘ leaves the cell with no format, and Excel files written that way fall back to libxlsxwriter’s own blue-underlined default.

`date_format`, `datetime_format`
[xl_format] number formats applied to ‘Date‘ / ‘POSIXct‘ values.

`date_col_width`, `datetime_col_width`
Default column width for ‘Date‘ / ‘POSIXct‘ columns.

`header_row_height`
Height (in points) of the header row.

Value

An ‘xl_properties‘ object.

Time zones

Excel has no concept of a time zone. When every ‘POSIXct‘ in the workbook shares one time zone, writexl drops the zone and writes local wall-clock time, and the default ‘datetime_format‘ loses its ‘" UTC"‘ suffix so that nothing is mislabelled. When the time zones differ, all datetimes are converted to UTC with a warning. Supplying your own ‘datetime_format‘ overrides the label in either case.

See Also

[xl_workbook], [write_xlsx]

Other workbook settings: `write_xlsx()`, `xl_workbook()`

Examples

```
xl_properties(title = "Quarterly report", author = "Finance",
              header_format = xl_font(bold = TRUE, color = "white") +
                              xl_fill(background = "navy"))
```

xl_rich_run *One run of a rich (multi-format) string*

Description

‘xl_rich_run()‘ is one fragment of an [xl_rich_string()]: a piece of text plus the font it is drawn in.

Usage

```
xl_rich_run(value, format = NULL)
```

Arguments

value	A single non-‘NA‘, non-empty string: the run’s text.
format	An optional [xl_format]. Only its **font** properties apply (see [xl_font()]); a run has no fill, border, alignment or number format, and supplying one warns. ‘NULL‘ draws the run in the cell’s own font.

Value

An ‘xl_rich_run‘ object.

See Also

[xl_rich_string], [xl_format]

Other cell content: `is_xl_comment()`, `xl_cell_general()`, `xl_comment()`, `xl_formula()`, `xl_rich_string()`

Examples

```
xl_rich_run("bold", xl_font(bold = TRUE))
xl_rich_run("plain")
```

xl_rich_string *A cell whose text has several formats*

Description

‘xl_rich_string()‘ builds the value of a single cell out of differently formatted runs, so that one cell can read "This is **bold** text". Pass it as the ‘value‘ of [xl_cell_general()].

Excel requires at least two runs: a string with one format is an ordinary character value, so pass it as one.

‘as.character()‘ returns the cell’s text with the per-run fonts dropped.

Usage

```
xl_rich_string(...)

is_xl_rich_string(x)

## S3 method for class 'xl_rich_string'
as.character(x, ...)
```

Arguments

...	Runs, in order. A bare string is taken as an unformatted run; an [xl_rich_run()] carries its own font. Lists of either are flattened, so runs can be assembled programmatically.
x	An object to test.

Value

An ‘xl_rich_string‘ object: a list of runs.

See Also

[xl_rich_run], [xl_cell_general], [xl_font]

Other cell content: `is_xl_comment()`, `xl_cell_general()`, `xl_comment()`, `xl_formula()`, `xl_rich_run()`

Examples

```
xl_rich_string("This is ", xl_rich_run("bold", xl_font(bold = TRUE)), " text")

# in a cell, with a cell-wide format alongside the per-run fonts
xl_cell_general(
  value  = xl_rich_string("2 H", xl_rich_run("2", xl_font(script = "sub")), "O"),
  format = xl_align(horizontal = "center")
)
```

---

xl_sheet *A worksheet with formatting and layout options*

Description

‘xl_sheet()‘ wraps a data frame together with worksheet-level options (column/row formatting and geometry, frozen panes, gridlines, tab color, zoom). Pass it anywhere [write_xlsx()] accepts a data frame; a plain data frame continues to behave exactly as before.

Usage

```
xl_sheet(
  data,
  cols = NULL,
  rows = NULL,
  freeze = NULL,
  gridlines = NA,
  tab_color = NA,
  zoom = NA,
  default_row_height = NA,
  auto_colwidth = FALSE,
  autofilter = FALSE,
  protect = FALSE,
  comment_author = NA,
  show_comments = FALSE,
  page = NULL,
  view = NULL,
  merge = NULL,
  validation = NULL,
  conditional = NULL,
  filter = NULL,
  outline = NULL,
  ignore_errors = NULL,
  table = NULL,
  image = NULL,
  background_image = NULL,
  chart = NULL
)
```

Arguments

data	A data frame (the sheet contents).
cols	An [xl_col_spec()], or a list of them.
rows	An [xl_row_spec()], or a list of them.
freeze	Frozen panes: an Excel cell reference such as ‘"A2"‘ (freeze the rows above and columns left of that cell), or ‘list(row =, col =)‘ giving the number of rows/columns to freeze.
gridlines	Logical; show (‘TRUE‘) or hide (‘FALSE‘) screen gridlines. ‘NA‘ leaves Excel’s default.
tab_color	Sheet tab color (see [xl_color]).
zoom	Zoom level as a percentage (10–400).
default_row_height	Default height (in points) for rows in the sheet.
auto_colwidth	If ‘TRUE‘, size each column to fit its contents (a character-count heuristic, since the xlsx format has no true "AutoFit"). Columns given an explicit width via [xl_col_spec()] are left untouched.

`autofilter` Add an autofilter (filter dropdowns). ‘TRUE‘ covers the whole used range (header plus data); an Excel range string such as ‘"A1:D51"‘ restricts it; ‘FALSE‘ (default) adds none.

`protect` Protect the worksheet: ‘FALSE‘ (default) leaves it unprotected, ‘TRUE‘ applies the standard protection, and a string sets a password. A named list gives fine-grained control, e.g. ‘list(password = "secret", format_cells = TRUE)‘; an editing option set to ‘TRUE‘ *allows* that action on the protected sheet. Available option names: ‘format_cells‘, ‘format_columns‘, ‘format_rows‘, ‘insert_columns‘, ‘insert_rows‘, ‘insert_hyperlinks‘, ‘delete_columns‘, ‘delete_rows‘, ‘sort‘, ‘autofilter‘, ‘pivot_tables‘, ‘scenarios‘, ‘objects‘, ‘no_select_locked_cells‘, ‘no_select_unlocked_cells‘. Cell locking via [xl_protection()] only has an effect on a protected sheet.

`comment_author` Default author for this sheet’s cell comments (a per-comment ‘author‘ overrides it).

`show_comments` If ‘TRUE‘, all comments on the sheet are initially shown (individual comments can still be forced via ‘xl_comment(visible=)‘).

`page` An [xl_page_setup()] describing how the sheet prints (orientation, paper size, margins, scaling, header and footer). Affects printing only, never the cell data.

`view` An [xl_sheet_view()] describing the sheet’s tab state and opening view (active/selected/hidden tab, selection, scroll position, zero display, direction, split panes).

`merge` One [xl_merge()], or a list of them, merging rectangles of cells into single cells. Merges are applied after the sheet’s rows are written, so a merge over cells the data frame filled keeps only the merged text — as merging in Excel does. Any merge turns off the memory-efficient row-streaming mode, since it writes back over rows already emitted.

`validation` One [xl_validation()], or a list of them, restricting what may be typed into a range — a dropdown, a numeric or date bound, a text length limit or a custom formula.

`conditional` One conditional format ([xl_cond_cell()], [xl_cond_scale()], [xl_cond_bar()], [xl_cond_icons()]), or a list of them, formatting cells according to their contents.

`filter` One [xl_filter()], or a list of them, setting autofilter criteria. Each also hides the rows it excludes, because Excel does not apply a filter when a file is opened — criteria on their own produce a sheet that looks filtered but shows every row. Implies ‘autofilter = TRUE‘ over the used range when ‘autofilter‘ is not set separately.

`outline` An [xl_outline()] controlling how the grouping symbols created by ‘level‘ in [xl_col_spec()] / [xl_row_spec()] are drawn. It changes their display only, never which rows are grouped.

`ignore_errors` A named list turning off the green error triangle Excel shows in cells it believes are wrong. Each name is an error type and each value a range, e.g. ‘list(number_stored_as_text = "A2:A99")‘. Types: ‘number_stored_as_text‘, ‘eval_error‘, ‘formula_differs‘, ‘formula_range‘, ‘formula_unlocked‘, ‘empty_cell_reference‘, ‘list_data_validation‘, ‘calculated_column‘, ‘two_digit_text_year‘.

`table` One [xl_table()], or a list of them, turning a range into an Excel table — a named, styled block with banded rows, a filter dropdown and an optional total row. Any table turns off the memory-efficient row-streaming mode, which libxlsxwriter refuses to combine with tables.

`image` One [xl_image()], or a list of them, placing images on the sheet — floating over the cells, or inside a cell with 'embed = TRUE'.

`background_image` An image tiled behind the sheet's cells, in any shape [xl_image()] accepts. It is a screen backdrop only — Excel never prints it.

`chart` One [xl_chart()], or a list of them, placed on the sheet and anchored to a cell. A chart's series may plot data from any sheet in the workbook, not only this one.

Value

An 'xl_sheet' object.

See Also

[xl_col_spec], [xl_row_spec], [write_xlsx]

Other worksheet layout: `xl_colrow_spec`, `xl_outline()`, `xl_page_setup()`, `xl_sheet_view()`

Examples

```
df <- data.frame(name = c("a", "b"), revenue = c(1000.5, 2000.25))
sheet <- xl_sheet(
  df,
  cols   = xl_col_spec("revenue", width = 14, format = xl_num_format("#,##0.00")),
  freeze = "A2",
  tab_color = "steelblue"
)
tmp <- write_xlsx(list(Data = sheet))
```

---

xl_sheet_view *How a worksheet appears when it opens*

Description

'xl_sheet_view()' collects a worksheet's tab state and opening view: which tab is active, selected or hidden, where the sheet is scrolled and selected, and a few display options. Pass it as 'xl_sheet(view =)'.

None of these affect the cell data.

Usage

```
xl_sheet_view(
  active = NA,
  selected = NA,
  visible = NA,
  first_tab = NA,
  selection = NULL,
  top_left = NULL,
  hide_zero = NA,
  right_to_left = NA,
  split = NULL
)
```

Arguments

active	Logical; make this the tab Excel opens on. At most one sheet in a workbook may be active.
selected	Logical; include this tab in the selected group. The active sheet is always selected.
visible	Logical; 'FALSE' hides the sheet's tab. A hidden sheet cannot be active or selected, the first sheet cannot be hidden unless another is made active, and at least one sheet must stay visible or Excel will not open the file. All four rules are checked before writing.
first_tab	Logical; make this the leftmost visible tab in the tab strip. Independent of which sheet is active.
selection	The cell or range selected when the sheet opens, as an Excel reference ('"B2"', '"B2:D10"') or a 'list(rows = , cols =)' spec. Excel also uses the order of a selection's corners to mark which cell in it is active; writexl does not expose that, because ranges are normalised by the shared range parser, which rejects an inverted range.
top_left	The cell scrolled to the top-left of the window when the sheet opens, as an Excel reference such as '"A5"'.
hide_zero	Logical; display zero values as blank cells.
right_to_left	Logical; order the columns right to left, for a sheet in a right-to-left language.
split	Split the sheet into scrollable panes with a visible, movable divider, given as the cell reference the split sits above and to the left of — '"B3"' splits above row 3 and left of column B. Mutually exclusive with 'xl_sheet(freeze =)', which does the same thing without the divider. libxlsxwriter positions a split by distance, in row-height and column-width units, not by row and column number. writexl converts the cell reference using the sheet's actual row heights and column widths, so the split lands where you asked even after resizing. Pass 'list(vertical = , horizontal =)' to give those units directly. Note that libxlsxwriter derives the pane's scroll anchor back from that distance assuming default row heights, so on a sheet with resized rows or columns the divider is placed correctly but the anchor cell may be a row or two out.

Value

An ‘xl_sheet_view‘ object.

See Also

[xl_sheet], [xl_page_setup]

Other worksheet layout: xl_colrow_spec, xl_outline(), xl_page_setup(), xl_sheet()

Examples

```
xl_sheet_view(active = TRUE, selection = "B2")
xl_sheet_view(visible = FALSE)

df <- data.frame(x = 1:3)
tmp <- write_xlsx(list(
  Summary = xl_sheet(df, view = xl_sheet_view(active = TRUE)),
  Working = xl_sheet(df, view = xl_sheet_view(visible = FALSE))
))
```

---

xl_table *Add a worksheet table*

Description

‘xl_table()‘ turns a range into an Excel table: a named, styled block with banded rows, a filter dropdown in its header, and an optional total row. Pass one or a list of them as ‘xl_sheet(table =)‘.

Column headers default to the data frame’s column names. That is not just a convenience: Excel records a table’s column names separately from the header cells and rejects a file where the two disagree, so the default is what makes a table safe to add at all.

Usage

```
xl_table(
  range = NULL,
  name = NULL,
  style = "medium 9",
  header_row = TRUE,
  autofilter = TRUE,
  banded_rows = TRUE,
  banded_columns = FALSE,
  first_column = FALSE,
  last_column = FALSE,
  total_row = FALSE,
  columns = NULL
)
```

Arguments

`range`	The cells the table covers, as an Excel range string or a ‘list(rows = , cols =)‘ spec. Defaults to the sheet’s used range, including the header row and, with ‘total_row = TRUE‘, one row below.
`name`	The table’s name. See "Table names".
`style`	The table style: ‘"none"‘, or a type and number such as ‘"medium 9"‘ (the default), ‘"light 21"‘ or ‘"dark 11"‘. Light styles are numbered 0–21, medium 1–28 and dark 1–11.
`header_row`	Show the header row. Turning it off also removes the filter dropdown, as it does in Excel.
`autofilter`	Show the filter dropdown in the header row.
`banded_rows, banded_columns`	Alternating row / column shading.
`first_column, last_column`	Highlight the first / last column.
`total_row`	Add a total row below the data. See "The total row".
`columns`	One [xl_table_column()], or a list of them, overriding individual columns.

Value

An ‘xl_table‘ object.

Table names

Excel requires table names to be unique across the workbook, and formulas refer to a table by name. ‘name = NULL‘ (the default) has writexl generate a unique name from the sheet name and write it explicitly, so it cannot shift when another table is added elsewhere. Give a string to choose your own; it is validated and checked for collisions. Give ‘NA‘ to write no name at all and let Excel assign ‘Table1‘, ‘Table2‘, ... by insertion order — which makes any formula naming the table fragile, so that combination warns.

The total row

A total row is written by libxlsxwriter below the data, and the sheet’s row plan does not know about it: ‘auto_colwidth‘ does not measure it, and an [xl_row_spec()] aimed at that row will fight it. It has no cells of its own until a column gives a ‘total‘ or ‘total_label‘.

See Also

[xl_table_column], [xl_sheet]

Other worksheet features: `xl_conditional`, `xl_filter()`, `xl_filter_keep()`, `xl_merge()`, `xl_table_column()`, `xl_validation()`

Examples

```
xl_table(style = "light 9")
xl_table(name = "Sales", total_row = TRUE,
         columns = list(xl_table_column("fruit", total_label = "Total"),
                        xl_table_column("qty", total = "sum")))
```

`xl_table_column` *Describe a column of a worksheet table*

Description

‘xl_table_column()‘ overrides what [xl_table()] does with one column: its header caption, a formula filling the column, what its total-row cell shows, and the formats applied to the header and the data cells.

Only the columns you name need an entry; the rest take the data frame’s column name as their header and are left otherwise alone.

Usage

```
xl_table_column(
  col,
  header = NULL,
  formula = NULL,
  total = NULL,
  total_label = NULL,
  total_value = NULL,
  format = NULL,
  header_format = NULL
)
```

Arguments

`col`	The column: a name or a 1-based position within the table’s range.
`header`	The header caption. Defaults to the data frame’s column name — which is also what makes a table safe to add, since Excel rejects a file whose table definition and header cells disagree.
`formula`	A formula filling every data cell of the column, usually with a structured reference such as ‘"=SUM(Sales[@[Q1]:[Q4]])"‘. Because those name the table, see the ‘name‘ argument of [xl_table()].
`total`	The function shown in this column’s total-row cell: one of ‘"sum"‘, ‘"average"‘, ‘"count"‘, ‘"count_nums"‘, ‘"max"‘, ‘"min"‘, ‘"std_dev"‘ or ‘"var"‘. Needs ‘total_row = TRUE‘ on the table.
`total_label`	A string for the total-row cell instead of a function — typically ‘"Total"‘ under the first column.

total_value	The number the total-row cell already holds. Excel recalculates ‘total‘ on open and ignores this, but a reader that does not evaluate formulas — [readxl::read_xlsx()] among them — shows what is cached, which without this is nothing.
format	An [xl_format] applied to the column’s data cells.
header_format	An [xl_format] applied to the column’s header cell.

Value

An ‘xl_table_column‘ object.

See Also

[xl_table], [xl_sheet]

Other worksheet features: xl_conditional, xl_filter(), xl_filter_keep(), xl_merge(), xl_table(), xl_validation()

Examples

```
xl_table_column("qty", total = "sum")
xl_table_column("fruit", total_label = "Total")
xl_table_column("margin", formula = "=Sales[@revenue] * 0.3")
```

---

xl_validation *Restrict what can be typed into a range*

Description

‘xl_validation()‘ adds Excel data validation to a range: a dropdown list, a numeric or date bound, a text-length limit, or a custom formula. Pass one or a list of them as ‘xl_sheet(validation =)‘.

Excel’s 17 internal validation types are collapsed into the five ‘type‘ kinds below. Whether a limit is a literal, a cell formula or a date is inferred from what you pass: a string starting with ‘"="‘ is a formula, and a ‘Date‘ or ‘POSIXct‘ is a date/time bound.

Usage

```
xl_validation(
  range,
  type = "any",
  criteria = NA,
  value = NULL,
  min = NULL,
  max = NULL,
  list = NULL,
  input_title = NA,
  input_message = NA,
  error_title = NA,
  error_message = NA,
```

```
  error_type = NA,
  ignore_blank = NA,
  show_input = NA,
  show_error = NA,
  dropdown = NA
)
```

Arguments

range	The cells to validate: an Excel range string such as ‘"B2:B100"‘, a single cell, or a ‘list(rows = , cols =)‘ spec.
type	The kind of value allowed: ‘"integer"‘, ‘"decimal"‘, ‘"date"‘, ‘"time"‘, ‘"length"‘ (of the text entered), ‘"custom"‘ (any formula that must evaluate ‘TRUE‘), or ‘"any"‘ (no restriction, useful when you only want the input message). Ignored when ‘list‘ is given.
criteria	How ‘value‘ (or ‘min‘/‘max‘) limits the entry: ‘"between"‘, ‘"not between"‘, ‘"=="‘, ‘"!="‘, ‘">"‘, ‘"<"‘, ‘">="‘ or ‘"<="‘. Required for the numeric, date, time and length kinds; must not be given for ‘list‘, ‘custom‘ or ‘any‘, which carry their own meaning. Supplying ‘min‘ and ‘max‘ implies ‘"between"‘.
value	The single limit the criteria applies to. A number, a ‘Date‘ or ‘POSIXct‘, or a ‘"=..."‘ formula.
min, max	The two limits for ‘"between"‘ / ‘"not between"‘. Supplying both and omitting ‘criteria‘ implies ‘"between"‘.
list	A dropdown of allowed values: a character vector of choices, or a single ‘"=..."‘ formula naming a range that holds them. The choices are stored joined by commas, and Excel limits that joined string to 255 characters.
input_title, input_message	Text shown in a tooltip when the cell is selected. Titles are limited to 32 characters and messages to 255.
error_title, error_message	Text shown when an invalid entry is made. Same limits.
error_type	What Excel does on an invalid entry: ‘"stop"‘ (refuse it), ‘"warning"‘ or ‘"information"‘ (both allow it through).
ignore_blank	Logical; allow an empty cell. ‘TRUE‘ by default, as in Excel.
show_input, show_error	Logical; whether the input tooltip and the error alert are shown at all. Both ‘TRUE‘ by default.
dropdown	Logical; show the in-cell dropdown arrow for a ‘list‘ validation. ‘TRUE‘ by default.

Value

An ‘xl_validation‘ object.

See Also

[xl_sheet]

Other worksheet features: xl_conditional, xl_filter(), xl_filter_keep(), xl_merge(), xl_table(), xl_table_column()

Examples

```
# a dropdown
xl_validation("C2:C100", list = c("open", "high", "close"))

# a numeric bound, with the message Excel shows on a bad entry
xl_validation("B2:B100", type = "integer", min = 1, max = 10,
              error_message = "Enter a whole number from 1 to 10")

# a date bound
xl_validation("D2:D100", type = "date", criteria = ">=",
              value = as.Date("2024-01-01"))

df <- data.frame(qty = 1:3)
tmp <- write_xlsx(list(Data = xl_sheet(df,
  validation = xl_validation("A2:A4", type = "integer", min = 0, max = 99))))
```

---

xl_workbook *A workbook: sheets plus workbook-level properties*

Description

‘xl_workbook()‘ binds one or more sheets (data frames or [xl_sheet]s) to a set of [xl_properties]. It is the single place to attach workbook-level formatting defaults and metadata. When passed to [write_xlsx()], the workbook’s ‘col_names‘ and ‘format_headers‘ settings take precedence over ‘write_xlsx()‘’s own arguments.

Usage

```
xl_workbook(
  sheets,
  properties = xl_properties(),
  col_names = TRUE,
  format_headers = TRUE
)
```

Arguments

sheets	A data frame, an [xl_sheet], or a (named) list of them.
properties	An [xl_properties] object.
col_names	write column names as the header row at the top of the sheet?
format_headers	apply the workbook’s header format to that header row? The default header format is bold and centered; change it with xl_properties(header_format =).

Value

An ‘xl_workbook‘ object.

See Also

[xl_properties], [xl_sheet], [write_xlsx]

Other workbook settings: `write_xlsx()`, `xl_properties()`

Examples

```
wb <- xl_workbook(
  list(Data = data.frame(x = 1:3)),
  properties = xl_properties(title = "Demo", author = "me")
)
tmp <- write_xlsx(wb)
```

Index