

Package: openxlsx2Extras (via r-universe)

October 16, 2026

Title Extra Functions for the openxlsx2 package

Version 0.0.0.9000

Description Extends the openxlsx2 package with wrapper and helper functions designed to add new features and options when working with Excel workbooks.

License MIT + file LICENSE

URL <https://github.com/elipousson/openxlsx2Extras>,
<https://elipousson.github.io/openxlsx2Extras/>

BugReports <https://github.com/elipousson/openxlsx2Extras/issues>

Depends R (>= 4.1.0)

Imports cli, fs, lifecycle, openxlsx2 (>= 1.23), purrr, rlang, utils,
vctrs

Suggests dplyr, gt, marquee, sf, testthat (>= 3.0.0), tidyselect,
withr

Config/testthat/edition 3

Encoding UTF-8

Language en-US

Roxygen list(markdown = TRUE)

Config/roxygen2/version 8.1.0

Config/pak/sysreqs cmake make libicu-dev libuv1-dev

Repository <https://elipousson.r-universe.dev>

Date/Publication 2026-09-16 18:17:05 UTC

RemoteUrl <https://github.com/elipousson/openxlsx2Extras>

RemoteRef HEAD

RemoteSha 75510a4d97164005aa36ec082dd86d6178beb9ec

Contents

as_data_validation_value	2
as_wb	3
common_dxfs_styles	5
convert_to	5
fnt_lgl_cols	7
fnt_marquee_txt	8
prep_wb_data	9
read_xlsx_ext	10
set_excel_fmt_class	12
vec_fmt_lgl	13
wb_add_data_ext	14
wb_add_marquee_text	16
wb_add_styles	18
wb_new_workbook	18
wb_protect_worksheets	20
wb_rename_data	21
wb_rename_sheets	22
wb_save_ext	23
wb_set_properties_ext	24
wb_split	25
wb_to_df_list	26
write_xlsx_ext	27
Index	30

as_data_validation_value	<i>Prepare a character vector as a data validation string</i>
--------------------------	---

Description

`as_data_validation_value()` collapses a character vector as a single string for use as a validation value argument by `openxlsx2::wb_add_data_validation()` when `type = "list"`.

Usage

```
as_data_validation_value(x, allow_blank = TRUE)
```

Arguments

- `x` A character vector of options to allow in cell validation.
- `allow_blank` If TRUE, add a blank space " " to the returned values.

Value

A length-1 character string with `x` collapsed into a comma-separated, quoted list.

Examples

```
as_data_validation_value(c("Option A", "Option B", "Option C"))
```

as_wb	<i>Coerce a data frame or list of data frames to a workbook</i>
-------	---

Description

`as_wb()` converts a data frame, a list of data frames, or an Excel file path to a `wbWorkbook` object.

Usage

```
as_wb(
  x,
  ...,
  sheet_names = NULL,
  creator = NULL,
  title = NULL,
  subject = NULL,
  category = NULL,
  datetime_created = Sys.time(),
  theme = NULL,
  keywords = NULL,
  properties = NULL,
  call = caller_env()
)
```

Arguments

<code>x</code>	A data frame, a list of data frames, a file path for an Excel file, or a <code>wbWorkbook</code> object. A <code>wbWorkbook</code> is returned "as is" ignoring all other parameters. A file path is loaded to a data frame using <code>openxlsx2::wb_to_df()</code> .
<code>...</code>	Arguments passed on to <code>wb_add_data_ext</code>
<code>as_table</code>	Default FALSE. If TRUE, use <code>openxlsx2::wb_add_data_table()</code> to add data to workbook. If FALSE, use <code>openxlsx2::wb_add_data()</code> . Additional parameters in <code>...</code> are passed to one function or the other depending on this value.
<code>labels</code>	Method for handling column labels. "drop" (default), "row_before", or "comments". If "row_before", insert column labels in the row before the column names. If "comments", add column labels as columns on the column names in the start row.
<code>na.strings</code>	[Deprecated] Use <code>na</code> instead.
<code>list_columns</code>	String, one of "collapse" (default), "drop", or "asis"

sep	String to use in collapsing list columns. Ignored unless <code>list_columns = "collapse"</code> . Defaults to <code>"; "</code> .
geometry	String, one of "drop" (default), "coords", or "wkt". "coords" uses <code>sf::st_centroid()</code> to convert input to POINT geometry, transforms geometry to EPSG:4326, converts geometry to coordinates, and adds new columns with names matching <code>coords</code> . "wkt" converts geometry to a Well Known Text (WKT) character vector using <code>sf::st_as_text()</code> and replaces the existing geometry column (keeping the existing sf column name).
coords	Length 2 character vector with column names to add if <code>geometry = "coords"</code> . Must be length 2 in longitude, latitude order.
wb	A Workbook object containing a worksheet.
sheet	The worksheet to write to. Can be the worksheet index or name.
start_row	A vector specifying the starting row to write <code>x</code> to.
na	Value used for replacing NA values from <code>x</code> . Default looks if <code>options("openxlsx2.na")</code> is set. Otherwise <code>na_strings()</code> uses the special #N/A value within the workbook.
sheet_names	Optional character vector of worksheet names.
creator	Creator of the workbook (your name). Defaults to login username or <code>options("openxlsx2.creator")</code> if set.
title, subject, category, keywords	Additional arguments passed to <code>openxlsx2::wb_workbook()</code> .
datetime_created	The time of the workbook is created
theme	Optional theme identified by string or number. See Details for options.
properties	A named list (typically from <code>openxlsx2::wb_get_properties()</code>) used to set new workbook properties for any values set to NULL. <code>datetime_created</code> defaults to <code>Sys.time()</code> so must be set to NULL to inherit value from <code>properties</code> .
call	The execution environment of a currently running function, e.g. <code>caller_env()</code> . The function will be mentioned in error messages as the source of the error. See the <code>call</code> argument of <code>abort()</code> for more information.

Value

A `wbWorkbook` object.

See Also

- `write_xlsx_ext()`
- `map_wb()`

Examples

```
as_wb(mtcars[1:3, ])

as_wb(list(mtcars[1:3, ], mtcars[4:6, ]))
```

common_dxfs_styles	<i>Common dxfs styles</i>
--------------------	---------------------------

Description

`common_dxfs_styles` is a named list of dxfs styles created with `openxlsx2::create_dxfs_style()` with bad, good, and neutral values.

Usage

```
common_dxfs_styles
```

Source

https://stackoverflow.com/questions/27611260/what-are-the-rgb-codes-for-the-conditional-formatting-styles-in-excel#comment78058968_27611522

See Also

`wb_add_styles()`

convert_to	<i>Convert CSV to XLSX files or XLSX to CSV</i>
------------	---

Description

A set of functions to convert between CSV and XLSX formats using flexible input and output options.

Usage

```
csv_to_wb(file, new_file = NULL, .f = utils::read.csv, ...)  
  
csv_to_xlsx(file, new_file = NULL, .f = utils::read.csv, ...)  
  
xlsx_to_csv(  
  file,  
  new_file = NULL,  
  sheet = 1,  
  .f = utils::write.csv,  
  ext = "csv",  
  ...  
)
```

Arguments

file	Path or paths to input files. For <code>csv_to_wb()</code> , users can pass multiple CSV files when creating a workbook or xlsx file. If these inputs are named, the names are used as worksheet names.
new_file	Path to output file. Optional. If new_file is not supplied and file is a string, new_file is set to use the same path with a new file extension or (if file is not a string) new_file is set to a temporary file.
.f	Function used to read or write the csv file. Defaults to <code>utils::read.csv</code> for <code>csv_to_wb()</code> and <code>csv_to_xlsx()</code> and <code>utils::write.csv</code> for <code>xlsx_to_csv()</code> . Other functions are allowed but must use the input or output file name as the second argument.
...	Additional arguments passed to .f
sheet	A sheet in the workbook specified by file (either an index or a sheet name). Defaults to 1.
ext	File extension for output file. Defaults to "csv".

Details

These functions allow seamless conversion between CSV and XLSX formats:

- `csv_to_wb`: Reads one or more CSV files and writes them to a workbook object.
- `csv_to_xlsx`: Converts one or more CSV files to a XLSX file.
- `xlsx_to_csv`: Converts an XLSX file to a CSV file.

Value

A `wbWorkbook` object.

A `wbWorkbook` object, saved to **new_file** as a side effect.

The return value of **.f**, typically the invisible NULL result of `utils::write.csv()`. The converted data is written to **new_file** as a side effect.

Author(s)

Jordan Mark Barbone <jmbarbone@gmail.com>

Jan Marvin Garbuszus <jan.garbuszus@ruhr-uni-bochum.de>

Eli Pousson <eli.pousson@gmail.com>

Examples

```
# Create example CSV file
csv <- tempfile(fileext = ".csv")
utils::write.csv(x = mtcars, file = csv)

# Convert CSV to Workbook
wb <- csv_to_wb(file = csv)

# Convert CSV to XLSX
```

```

xlsx <- openxlsx2::temp_xlsx()
csv_to_xlsx(file = csv, new_file = xlsx)

# Convert XLSX back to CSV
xlsx_to_csv(file = xlsx, new_file = csv)

```

fmt_lgl_cols

Format logical vector columns to use replacement values

Description

`fmt_lgl_cols()` uses `vec_fmt_lgl()` to format all (or specified) logical vector columns to use replacement values.

Usage

```

fmt_lgl_cols(
  .data,
  .cols = tidyselect::where(is.logical),
  values = c("Y", "N")
)

```

Arguments

<code>.data</code>	A data frame, data frame extension (e.g. a tibble), or a lazy data frame (e.g. from <code>dbplyr</code> or <code>dtplyr</code>). See <i>Methods</i> , below, for more details.
<code>.cols</code>	<tidy-select> Columns to transform. You can't select grouping columns because they are already automatically handled by the verb (i.e. <code>summarise()</code> or <code>mutate()</code>).
<code>values</code>	Default to <code>c("Y", "N")</code> Length 2 vector where first element replaces <code>TRUE</code> and the second element replaces <code>FALSE</code> .

Value

A data frame matching `.data` with logical columns replaced by character values.

Examples

```
fmt_lgl_cols(data.frame(x = c(TRUE, FALSE, TRUE)))
```

fmt_marquee_txt	Use <code>marquee::marquee_parse()</code> to format Markdown text
-----------------	---

Description

[Experimental]

`fmt_marquee_txt()` uses `marquee::marquee_parse()` and `openxlsx2::fmt_txt()` to format Markdown text.

Usage

```
fmt_marquee_txt(
  text,
  ...,
  style = marquee::classic_style(),
  ignore_html = TRUE
)
```

Arguments

<code>text</code>	A character string. The core quality of markdown is that any text is valid markdown so there is no restrictions on the content
<code>...</code>	Ignored at this time. For future use.
<code>style</code>	A style set such as <code>classic_style()</code> that defines how the text should be rendered
<code>ignore_html</code>	Should HTML code be removed from the output

Value

A list of `fmt_txt` objects, one per parsed line of `text`.

Examples

```
fmt_marquee_txt(
  "# ABC

  abc"
)
```

prep_wb_data	<i>Prepare data for adding to a workbook</i>
--------------	--

Description

`prep_wb_data()` prepares a data frame for addition to a workbook by handling list columns and geometry columns (for sf data frames).

Usage

```
prep_wb_data(
  x,
  list_columns = c("collapse", "drop", "asis"),
  sep = "; ",
  geometry = c("drop", "coords", "wkt"),
  coords = c("lon", "lat"),
  call = caller_env()
)
```

Arguments

x	Required. A data frame or an object coercible to a data frame with <code>base::as.data.frame()</code> .
list_columns	String, one of "collapse" (default), "drop", or "asis"
sep	String to use in collapsing list columns. Ignored unless <code>list_columns = "collapse"</code> . Defaults to "; ".
geometry	String, one of "drop" (default), "coords", or "wkt". "coords" uses <code>sf::st_centroid()</code> to convert input to POINT geometry, transforms geometry to EPSG:4326, converts geometry to coordinates, and adds new columns with names matching <code>coords</code> . "wkt" converts geometry to a Well Known Text (WKT) character vector using <code>sf::st_as_text()</code> and replaces the existing geometry column (keeping the existing sf column name).
coords	Length 2 character vector with column names to add if <code>geometry = "coords"</code> . Must be length 2 in longitude, latitude order.
call	The execution environment of a currently running function, e.g. <code>caller_env()</code> . The function will be mentioned in error messages as the source of the error. See the <code>call</code> argument of <code>abort()</code> for more information.

Value

A data frame with list and geometry columns handled as specified.

Examples

```
list_df <- vctrs::data_frame(
  num = 1,
  alpha = list(list("A", "B", "C"))
)

prep_wb_data(list_df)

prep_wb_data(list_df, list_columns = "drop")

prep_wb_data(list_df, list_columns = "asis")

if (rlang::is_installed("sf")) {
  nc <- sf::read_sf(system.file("shape/nc.shp", package = "sf"))

  prep_wb_data(nc, geometry = "coords")

  prep_wb_data(nc, geometry = "wkt")
}
```

read_xlsx_ext

Create a data frame from a Workbook (with extra features)

Description

[Experimental]

`read_xlsx_ext()` uses `openxlsx2::read_xlsx()` with a few added features:

- allows use of a name repair argument ("`unique_quite`" by default) to avoid blank "" or NA values for column names.
- allows vector inputs for the file or sheet argument. These parameters are recycled to a common length and result in the return of a data frame list unless the `combine = TRUE` is set. If `combine = TRUE`, set `names_to` (passed to `purrr::list_rbind`) to combined the file basename values (default) or full path values as a column (depending on the `names_from` argument). `names_from` can also be a length > 1 character vector that can be recycled to match the length of file.

Usage

```
read_xlsx_ext(
  file,
  sheet = 1,
  ...,
  names_from = "basename",
  names_to = rlang::zap(),
  combine = TRUE,
```

```
    repair = "unique_quiet"
  )
```

Arguments

file	A workbook file path, a wbWorkbook object, or a valid URL.
sheet	Defaults to 1.
...	Arguments passed on to openxlsx2::read_xlsx
start_row, start_col	Optional numeric values specifying the first row or column to begin data discovery.
row_names	Logical; if TRUE, uses the first column of the selection as row names.
col_names	Logical; if TRUE, uses the first row of the selection as column headers.
skip_empty_rows, skip_empty_cols	Logical; if TRUE, filters out rows or columns containing only missing values.
skip_hidden_rows, skip_hidden_cols	Logical; if TRUE, excludes rows or columns marked as hidden in the worksheet metadata.
rows, cols	Optional numeric vectors specifying the exact indices to read.
detect_dates	Logical; if TRUE, identifies date and datetime styles for conversion.
na	A character vector or a named list (e.g., <code>list(strings = "", numbers = -99)</code>) defining values to treat as NA.
fill_merged_cells	Logical; if TRUE, propagates the top-left value of a merged range to all cells in that range.
dims	A character string defining the range. Supports wildcards (e.g., "A1:++" or "A-:+5").
show_formula	Logical; if TRUE, returns the formula strings instead of calculated values.
convert	Logical; if TRUE, attempts to coerce columns to appropriate R classes.
types	A named vector (numeric or character) to explicitly define column types.
named_region	A character string referring to a defined name or spreadsheet Table.
keep_attributes	Logical; if TRUE, attaches metadata such as the internal type table (tt) and types as attributes to the output.
check_names	Logical; if TRUE, ensures column names are syntactically valid R names via make.names() .
show_hyperlinks	Logical; if TRUE, replaces cell values with their underlying hyperlink targets.
apply_numfmts	Logical; if TRUE, applies spreadsheet number formatting and returns strings.
names_from	String, one of "basename" (default) or "path", used to name the elements of file . Can also be a character vector recycled to the length of file

	to supply custom names. These names are used as the list names when <code>combine = FALSE</code> and passed to <code>purrr::list_rbind()</code> as the source of <code>names_to</code> values when <code>combine = TRUE</code> .
<code>names_to</code>	By default, <code>names(x)</code> are lost. To keep them, supply a string to <code>names_to</code> and the names will be saved into a column with that name. If <code>names_to</code> is supplied and <code>x</code> is not named, the position of the elements will be used instead of the names.
<code>combine</code>	If <code>TRUE</code> , always return a data frames. If <code>FALSE</code> , return a list of data frames.
<code>repair</code>	Either a string or a function. If a string, it must be one of "check_unique", "minimal", "unique", "universal", "unique_quiet", or "universal_quiet". If a function, it is invoked with a vector of minimal names and must return minimal names, otherwise an error is thrown. • Minimal names are never NULL or NA. When an element doesn't have a name, its minimal name is an empty string. • Unique names are unique. A suffix is appended to duplicate names to make them unique. • Universal names are unique and syntactic, meaning that you can safely use the names as variables without causing a syntax error.

The "check_unique" option doesn't perform any name repair. Instead, an error is raised if the names don't suit the "unique" criteria.

The options "unique_quiet" and "universal_quiet" are here to help the user who calls this function indirectly, via another function which exposes `repair` but not `quiet`. Specifying `repair = "unique_quiet"` is like specifying `repair = "unique"`, `quiet = TRUE`. When the "*_quiet" options are used, any setting of `quiet` is silently overridden.

Value

A data frame, or (if `combine = FALSE`) a list of data frames.

Examples

```
xlsx <- openxlsx2::temp_xlsx()
openxlsx2::write_xlsx(mtcars, xlsx)

read_xlsx_ext(xlsx)
```

<code>set_excel_fmt_class</code>	<i>Prepare a data frame with Excel style class values for formatting by openxlsx2</i>
----------------------------------	---

Description

[Experimental]

Usage

```
set_excel_fmt_class(
  data,
  cols,
  fmt_class = "currency",
  multiple = TRUE,
  strict = FALSE
)
```

Arguments

<code>data</code>	A data frame with columns to format.
<code>cols</code>	Column names or numbers to modify.
<code>fmt_class</code>	Excel style class, one of: c("currency", "accounting", "hyperlink", "percentage", "scientific", "formula"). Length is recycled to match length of cols using <code>vecrs::vec_recycle()</code> .
<code>multiple</code>	Whether <code>arg</code> may contain zero or several values.
<code>strict</code>	If TRUE, error if any character values in cols are not included in the names of data. Default FALSE.

Details

`set_excel_fmt_class()` applies a style to each specified column. See the openxlsx2 documentation for more information on this feature: https://janmarvin.github.io/openxlsx2/articles/openxlsx2_style_manual.html#numfmts2

Value

A data frame matching data with `fmt_class` added to the class of the specified cols.

Examples

```
data <- data.frame(price = c(1, 2), pct = c(0.1, 0.2))

set_excel_fmt_class(data, cols = c("price", "pct"), fmt_class = c("currency", "percentage"))
```

<code>vec_fmt_lgl</code>	<i>Format a logical vector to specified values</i>
--------------------------	--

Description

Format a logical vector to specified values

Usage

```
vec_fmt_lgl(x, values = c("Y", "N"))
```

Arguments

x	A logical vector.
values	Default to <code>c("Y", "N")</code> Length 2 vector where first element replaces TRUE and the second element replaces FALSE .

Value

A character vector the same length as **x**.

Examples

```
vec_fmt_lgl(c(TRUE, FALSE))

vec_fmt_lgl(c(TRUE, FALSE), c("Yes", "No"))
```

wb_add_data_ext	<i>Add data using <code>openxlsx2::wb_add_data()</code> with extra features</i>
------------------------	---

Description**[Experimental]**

`wb_add_data_ext()` extends `openxlsx2::wb_add_data()` to add data to a workbook with special handling for input data with geometry or list columns (using `prep_wb_data()`) and labelled data.

Usage

```
wb_add_data_ext(
  wb,
  x,
  sheet = current_sheet(),
  ...,
  start_row = 1,
  list_columns = c("collapse", "drop", "asis"),
  sep = "; ",
  geometry = c("drop", "coords", "wkt"),
  coords = c("lon", "lat"),
  labels = c("drop", "row_before", "comments"),
  as_table = FALSE,
  na = openxlsx2::na_strings(),
  na.strings = lifecycle::deprecated(),
  call = caller_env()
)
```

Arguments

wb	A Workbook object containing a worksheet.
x	Required. A data frame or an object coercible to a data frame with <code>base::as.data.frame()</code> .
sheet	The worksheet to write to. Can be the worksheet index or name.
...	Arguments passed on to <code>openxlsx2::wb_add_data</code> , <code>openxlsx2::wb_add_data_table</code>
dims	Spreadsheet cell range that will determine start_col and start_row : "A1", "A1:B2", "A:B"
start_col	A vector specifying the starting column to write x to.
array	A bool if the function written is of type array
col_names	If TRUE, column names of x are written.
row_names	If TRUE, the row names of x are written.
with_filter	If TRUE, add filters to the column name row. NOTE: can only have one filter per worksheet.
name	The name of a named region if specified.
apply_cell_style	Should we write cell styles to the workbook
remove_cell_style	keep the cell style?
inline_strings	write characters as inline strings
enforce	enforce that selected dims is filled. For this to work, dims must match x
table_style	Any table style name or "none" (see <code>vignette("openxlsx2_style_manual")</code>)
table_name	Name of table in workbook. The table name must be unique.
first_column	logical. If TRUE, the first column is bold.
last_column	logical. If TRUE, the last column is bold.
banded_rows	logical. If TRUE, rows are color banded.
banded_cols	logical. If TRUE, the columns are color banded.
total_row	logical. With the default FALSE no total row is added.
params	list. Optional arguments passed to the data table creation.
start_row	A vector specifying the starting row to write x to.
list_columns	String, one of "collapse" (default), "drop", or "asis"
sep	String to use in collapsing list columns. Ignored unless list_columns = "collapse". Defaults to "; ".
geometry	String, one of "drop" (default), "coords", or "wkt". "coords" uses <code>sf::st_centroid()</code> to convert input to POINT geometry, transforms geometry to EPSG:4326, converts geometry to coordinates, and adds new columns with names matching coords . "wkt" converts geometry to a Well Known Text (WKT) character vector using <code>sf::st_as_text()</code> and replaces the existing geometry column (keeping the existing sf column name).
coords	Length 2 character vector with column names to add if geometry = "coords". Must be length 2 in longitude, latitude order.
labels	Method for handling column labels. "drop" (default), "row_before", or "comments". If "row_before", insert column labels in the row before the column names. If "comments", add column labels as columns on the column names in the start row.

<code>as_table</code>	Default FALSE. If TRUE, use <code>openxlsx2::wb_add_data_table()</code> to add data to workbook. If FALSE, use <code>openxlsx2::wb_add_data()</code> . Additional parameters in ... are passed to one function or the other depending on this value.
<code>na</code>	Value used for replacing NA values from <code>x</code> . Default looks if <code>options("openxlsx2.na")</code> is set. Otherwise <code>na_strings()</code> uses the special #N/A value within the workbook.
<code>na.strings</code>	[Deprecated] Use <code>na</code> instead.
<code>call</code>	The execution environment of a currently running function, e.g. <code>caller_env()</code> . The function will be mentioned in error messages as the source of the error. See the <code>call</code> argument of <code>abort()</code> for more information.

Value

A `wbWorkbook` object.

Examples

```
wb <- wb_new_workbook("mtcars")

wb_add_data_ext(wb, mtcars)

wb_add_data_ext(wb, mtcars, as_table = TRUE)
```

`wb_add_marquee_text` *Add Markdown formatted text to a Workbook*

Description

[Experimental]

`wb_add_marquee_text()` uses `fmt_marquee_txt()` to add Markdown formatted text to a Workbook. `openxlsx2::wb_set_col_widths()` is applied to the workbook to adjust widths for the specified columns. Set `widths = NULL` to disable this functionality.

Usage

```
wb_add_marquee_text(
  wb = NULL,
  text,
  sheet = current_sheet(),
  dims = NULL,
  ...,
  cols = 1,
  widths = "auto"
)
```

Arguments

wb	A Workbook object containing a worksheet.
text	A character string. The core quality of markdown is that any text is valid markdown so there is no restrictions on the content
sheet	The worksheet to write to. Can be the worksheet index or name.
dims	Spreadsheet cell range that will determine start_col and start_row : "A1", "A1:B2", "A:B"
...	Arguments passed on to fmt_marquee_txt
style	A style set such as classic_style() that defines how the text should be rendered
ignore_html	Should HTML code be removed from the output
cols	Indices of cols to set/remove column widths.
widths	Width to set cols to specified column width or "auto" for automatic sizing. widths is recycled to the length of cols . openxlsx2 sets the default width is 8.43, as this is the standard in some spreadsheet software. See Details for general information on column widths.

Value

A wbWorkbook object.

Examples

```
library(openxlsx2)

wb <- wb_workbook()
wb <- wb_add_worksheet(wb)

wb <- wb_add_marquee_text(
  wb,
  text = "
# Heading 1

Example text.

~~Strikethrough text~~

## Heading 2

- Bulleted list item 1
  - Nested bullet
- Bulleted list item 2"
)
```

wb_add_styles	<i>Add a named list of styles to a workbook</i>
---------------	---

Description

`wb_add_styles()` extends `openxlsx2::wb_add_style()` by allowing the addition of multiple styles as a named list.

Usage

```
wb_add_styles(wb, styles, style_names = NULL)
```

Arguments

<code>wb</code>	A <code>wbWorkbook</code> object.
<code>styles</code>	Required. List or style xml character, created by a <code>create_*()</code> function. Passed to <code>style</code> argument of <code>openxlsx2::wb_add_style()</code> .
<code>style_names</code>	Optional if <code>styles</code> is a named list.

Value

A `wbWorkbook` object.

Examples

```
wb <- openxlsx2::wb_workbook()
wb <- openxlsx2::wb_add_worksheet(wb)

style <- openxlsx2::create_dxfs_style(font_color = openxlsx2::wb_color(hex = "FF0000"))
wb_add_styles(wb, styles = style, style_names = "myred")
```

wb_new_workbook	<i>Create a new workbook and add named work sheets</i>
-----------------	--

Description

`wb_new_workbook()` is a convenience function wrapping `openxlsx2::wb_workbook()` and `openxlsx2::wb_add_worksheet()` to offer quick and easy setup for new workbooks.

Usage

```
wb_new_workbook(
  sheet_names = NULL,
  ...,
  default = "Sheet",
  creator = NULL,
  title = NULL,
  subject = NULL,
  category = NULL,
  datetime_created = Sys.time(),
  theme = NULL,
  keywords = NULL,
  properties = NULL,
  call = caller_env()
)
```

Arguments

sheet_names	Optional character vector of worksheet names.
...	Arguments passed on to openxlsx2::wb_add_worksheet
grid_lines	Logical; if FALSE , the worksheet grid lines are hidden.
row_col_headers	Logical; if FALSE , row numbers and column letters are hidden.
tab_color	The color of the worksheet tab. Accepts a wb_color() object, a standard R color name, or a hex color code (e.g., "#4F81BD").
zoom	The sheet zoom level as a percentage; a numeric value between 10 and 400. Values below 10 default to 10.
header, footer	Default character vectors of length three for the left, center, and right sections of the header or footer.
odd_header, odd_footer	Specific definitions for odd-numbered pages. Defaults to the values provided in header and footer .
even_header, even_footer	Specific definitions for even-numbered pages. Defaults to the values provided in header and footer .
first_header, first_footer	Specific definitions for the first page of the worksheet. Defaults to the values provided in header and footer .
visible	The visibility state of the sheet. One of "visible", "hidden", or "veryHidden".
has_drawing	<i>defunct</i>
paper_size	An integer code representing a standard paper size. Refer to wb_page_setup() for a complete list of codes.
orientation	The page orientation, either "portrait" or "landscape".
hdpi, vdpi	The horizontal and vertical DPI (dots per inch) for printing and rendering. Can be set globally via <code>options("openxlsx2.hdpi")</code> .
default	Default prefix to use for numbered sheets. Default values are used if sheet_names = NULL or if n_sheets is greater than the length of sheet_names . Defaults to "Sheet".

<code>creator</code>	Creator of the workbook (your name). Defaults to login username or <code>options("openxlsx.creator")</code> if set.
<code>title, subject, category, keywords</code>	Additional arguments passed to <code>openxlsx2::wb_workbook()</code> .
<code>datetime_created</code>	The time of the workbook is created
<code>theme</code>	Optional theme identified by string or number. See Details for options.
<code>properties</code>	A named list (typically from <code>openxlsx2::wb_get_properties()</code>) used to set new workbook properties for any values set to <code>NULL</code> . <code>datetime_created</code> defaults to <code>Sys.time()</code> so must be set to <code>NULL</code> to inherit value from <code>properties</code> .
<code>call</code>	The execution environment of a currently running function, e.g. <code>caller_env()</code> . The function will be mentioned in error messages as the source of the error. See the <code>call</code> argument of <code>abort()</code> for more information.

Value

A `wbWorkbook` object.

See Also

`as_wb()`

Examples

```
wb_new_workbook()

wb_new_workbook("Sheet 1")

wb_new_workbook(c("Data", "Analysis"))
```

`wb_protect_worksheets`

Protect one or more sheets in a workbook

Description

Protect one or more sheets in a workbook

Usage

```
wb_protect_worksheets(
  wb,
  sheet = NULL,
  protect = TRUE,
  password = NULL,
  properties = c("insertColumns", "insertRows", "deleteColumns", "deleteRows")
)
```

Arguments

wb	A workbook object
sheet	A name or index of a worksheet
protect	Whether to protect or unprotect the sheet (default=TRUE)
password	(optional) password required to unprotect the worksheet
properties	Defaults to c("insertColumns", "insertRows", "deleteColumns", "deleteRows")

Value

A wbWorkbook object.

Examples

```
wb <- as_wb(mtcars[1:3, ])

wb_protect_worksheets(wb)
```

wb_rename_data	<i>Rename column names in workbook data</i>
----------------	---

Description

Use `tidyselect::eval_rename()` to rename columns in workbook data.

Usage

```
wb_rename_data(wb, ..., sheet = 1, start_row = 1, start_col = 1)
```

Arguments

wb	A wbWorkbook object.
...	<new_name> = <old_name> pairs passed to <code>tidyselect::eval_rename()</code> specifying how to rename columns.
sheet	Sheet containing the column names to rename. Defaults to 1.
start_row	Row number containing the column names. Defaults to 1.
start_col	Column number where data starts. Defaults to 1.

Value

A wbWorkbook object.

Examples

```
wb <- as_wb(mtcars[1:3, c("mpg", "hp")])

wb_rename_data(wb, MPG = mpg, HP = hp)
```

<code>wb_rename_sheets</code>	<i>Rename workbook sheets using a tidyselect syntax</i>
-------------------------------	---

Description

`wb_rename_sheets()` and `wb_rename_sheets_with()` use the tidyselect package to rename sheets.

Usage

```
wb_rename_sheets(wb, ...)
```

```
wb_rename_sheets_with(wb, .fn, .sheets = tidyselect::everything())
```

Arguments

<code>wb</code>	A <code>wbWorkbook</code> object.
<code>...</code>	For <code>wb_rename_sheets()</code> , <code><new_name> = <old_name></code> pairs passed to <code>tidyselect::eval_rename()</code> specifying how to rename sheets.
<code>.fn</code>	Function used to transform the selected sheet names.
<code>.sheets</code>	Tidyselect expression selecting which sheets to rename. Defaults to all sheets.

Value

A `wbWorkbook` object.

A `wbWorkbook` object.

Examples

```
wb <- as_wb(list(a = mtcars[1:2, ], b = mtcars[3:4, ]))
wb_rename_sheets(wb, sales = "a", inventory = "b")

wb2 <- as_wb(list(a = mtcars[1:2, ], b = mtcars[3:4, ]))
wb_rename_sheets_with(wb2, \(x) paste0("sheet_", x))
```

wb_save_ext	<i>Save a workbook object to file while filling file name from assigned workbook title</i>
-------------	--

Description

[Experimental]

`wb_save_ext()` is a helper function extending `openxlsx2::wb_save()` by filling a missing file name with the workbook title and validating the file extension. This function is not stable and may change in the future.

Usage

```
wb_save_ext(wb, file = NULL, overwrite = TRUE, flush = FALSE)
```

Arguments

<code>wb</code>	A <code>wbWorkbook</code> object to write to file
<code>file</code>	A path to save the workbook to
<code>overwrite</code>	If <code>FALSE</code> , will not overwrite when <code>file</code> already exists.
<code>flush</code>	Experimental, streams the worksheet file to disk

Value

A `wbWorkbook` object, saved to `file` as a side effect.

Examples

```
withr::with_tempdir({  
  wb <- wb_new_workbook(  
    title = "Title used for output file",  
    sheet_name = "Sheet 1"  
  )  
  
  wb_save_ext(wb)  
  
  fs::dir_ls()  
})
```

wb_set_properties_ext

Set workbook properties

Description

`wb_set_properties_ext()` extends `openxlsx2::wb_set_properties()` by adding a `properties` argument that uses a named list to default default values for existing properties.

Usage

```
wb_set_properties_ext(
  wb,
  ...,
  creator = NULL,
  title = NULL,
  subject = NULL,
  category = NULL,
  datetime_created = NULL,
  datetime_modified = NULL,
  modifier = NULL,
  keywords = NULL,
  comments = NULL,
  manager = NULL,
  company = NULL,
  custom = NULL,
  properties = NULL
)
```

Arguments

<code>wb</code>	A Workbook object
<code>...</code>	Must be empty.
<code>creator</code>	Creator of the workbook (your name). Defaults to login username or <code>options("openxlsx2.creator")</code> if set.
<code>title</code> , <code>subject</code> , <code>category</code> , <code>keywords</code> , <code>comments</code> , <code>manager</code> , <code>company</code>	Workbook property, a string.
<code>datetime_created</code>	The time of the workbook is created
<code>datetime_modified</code>	The time of the workbook was last modified
<code>modifier</code>	A character string indicating who was the last person to modify the workbook
<code>custom</code>	A named vector of custom properties added to the workbook
<code>properties</code>	A named character vector or named list of properties used as default values for any other parameter that is not explicitly set.

Value

A workbook with modified properties.

Examples

```
wb <- openxlsx2::wb_workbook()

wb_set_properties_ext(wb, title = "My title", creator = "Jane Doe")
```

wb_split	<i>Use <code>dplyr::group_split</code> to split a workbook into a list of workbooks</i>
----------	---

Description

`wb_split()` uses `wb_to_df_list()` to extract the data frames from each sheet of a workbook and then split the data frames by a `.key` argument then convert each new list of data frames back into a `wbWorkbook` object.

Usage

```
wb_split(
  file,
  .by,
  ...,
  .keep = TRUE,
  properties = "inherit",
  wb_params = list()
)
```

Arguments

file	A workbook file path, a <code>wbWorkbook</code> object, or a valid URL.
.by	Passed to <code>dplyr::group_split()</code> .
...	Arguments passed on to <code>wb_to_df_list</code>
sheet_names	Character vector of sheet names. If not supplied, all sheet names from the supplied workbook are used.
.keep	Should the grouping columns be kept?
properties	If "inherit" (default) and <code>file</code> is a workbook, inherit the workbook list element properties from the existing workbook. <code>properties</code> can also be NULL or a named character vector, a named list, or a bare list of the same length as the number of groups defined using the <code>.by</code> argument.
wb_params	List of additional parameters to pass to <code>map_wb()</code> and <code>as_wb()</code> .

Value

A list of `wbWorkbook` objects.

Examples

```
wb <- as_wb(list(mtcars[1:3, ], mtcars[4:6, ]))

wb_split(wb, .by = carb)
```

<code>wb_to_df_list</code>	<i>Convert a workbook to a list of data frames</i>
----------------------------	--

Description

`wb_to_df_list()` uses `openxlsx2::wb_to_df()` to extract each sheet of a workbook object into a data frame. Additional parameters ... are recycled to match the length of sheet names.

Usage

```
wb_to_df_list(file, sheet_names = NULL, ...)
```

Arguments

<code>file</code>	A workbook file path, a <code>wbWorkbook</code> object, or a valid URL.
<code>sheet_names</code>	Character vector of sheet names. If not supplied, all sheet names from the supplied workbook are used.
<code>...</code>	Arguments passed on to <code>openxlsx2::wb_to_df</code>
<code>sheet</code>	The name or index of the worksheet to read. Defaults to the first sheet.
<code>start_row, start_col</code>	Optional numeric values specifying the first row or column to begin data discovery.
<code>row_names</code>	Logical; if TRUE, uses the first column of the selection as row names.
<code>col_names</code>	Logical; if TRUE, uses the first row of the selection as column headers.
<code>skip_empty_rows, skip_empty_cols</code>	Logical; if TRUE, filters out rows or columns containing only missing values.
<code>skip_hidden_rows, skip_hidden_cols</code>	Logical; if TRUE, excludes rows or columns marked as hidden in the worksheet metadata.
<code>rows, cols</code>	Optional numeric vectors specifying the exact indices to read.
<code>detect_dates</code>	Logical; if TRUE, identifies date and datetime styles for conversion.

na A character vector or a named list (e.g., `list(strings = "", numbers = -99)`) defining values to treat as NA.

fill_merged_cells Logical; if TRUE, propagates the top-left value of a merged range to all cells in that range.

dims A character string defining the range. Supports wildcards (e.g., "A1:++" or "A-:+5").

show_formula Logical; if TRUE, returns the formula strings instead of calculated values.

convert Logical; if TRUE, attempts to coerce columns to appropriate R classes.

types A named vector (numeric or character) to explicitly define column types.

named_region A character string referring to a defined name or spreadsheet Table.

keep_attributes Logical; if TRUE, attaches metadata such as the internal type table (tt) and types as attributes to the output.

check_names Logical; if TRUE, ensures column names are syntactically valid R names via `make.names()`.

show_hyperlinks Logical; if TRUE, replaces cell values with their underlying hyperlink targets.

apply_numfmts Logical; if TRUE, applies spreadsheet number formatting and returns strings.

Value

A list of data frame lists.

Examples

```
wb <- as_wb(list(mtcars[1:3, ], mtcars[4:6, ]))

wb_to_df_list(wb)

wb_to_df_list(wb, "Sheet 1")
```

write_xlsx_ext

Write data to an xlsx file with additional features

Description

[Experimental]

`write_xlsx_ext()` wraps `wb_add_data_ext()` to provide an equivalent to `openxlsx2::write_xlsx()` with additional features. Arguments passed to `openxlsx2::wb_workbook()` are ignored if `x` is a workbook instead of a data frame.

Usage

```
write_xlsx_ext(
  x,
  file = NULL,
  ...,
  sheet_names = NULL,
  creator = NULL,
  title = NULL,
  subject = NULL,
  category = NULL,
  datetime_created = Sys.time(),
  theme = NULL,
  keywords = NULL,
  as_table = FALSE,
  start_row = 1,
  geometry = "drop",
  labels = "drop",
  na = openxlsx2::na_strings(),
  na.strings = lifecycle::deprecated(),
  overwrite = TRUE,
  call = caller_env()
)
```

Arguments

x	Required. A <code>wbWorkbook</code> object, a data frame, or a bare list of data frames. <code>x</code> can also be any object coercible to a data frame (other than a bare list) by <code>base::as.data.frame()</code> . If <code>x</code> is a named list and <code>sheet_names</code> is supplied, the existing names for <code>x</code> are ignored.
file	A path to save the workbook to
...	additional arguments
sheet_names	Optional character vector of worksheet names.
creator	Creator of the workbook (your name). Defaults to login username or <code>options("openxlsx2.creator")</code> if set.
title, subject, category, keywords	Additional workbook properties passed to <code>wb_new_workbook()</code> . Ignored (with creator and title) if <code>x</code> is a workbook instead of a data frame.
datetime_created	The time of the workbook is created
theme	Optional theme identified by string or number. See Details for options.
as_table	Default <code>FALSE</code> . If <code>TRUE</code> , use <code>openxlsx2::wb_add_data_table()</code> to add data to workbook. If <code>FALSE</code> , use <code>openxlsx2::wb_add_data()</code> . Additional parameters in <code>...</code> are passed to one function or the other depending on this value.
start_row	A vector specifying the starting row to write <code>x</code> to.

<code>geometry</code>	String, one of "drop" (default), "coords", or "wkt". "coords" uses <code>sf::st_centroid()</code> to convert input to POINT geometry, transforms geometry to EPSG:4326, converts geometry to coordinates, and adds new columns with names matching <code>coords</code> . "wkt" converts geometry to a Well Known Text (WKT) character vector using <code>sf::st_as_text()</code> and replaces the existing geometry column (keeping the existing sf column name).
<code>labels</code>	Method for handling column labels. "drop" (default), "row_before", or "comments". If "row_before", insert column labels in the row before the column names. If "comments", add column labels as columns on the column names in the start row.
<code>na</code>	Value used for replacing NA values from <code>x</code> . Default looks if <code>options("openxlsx2.na")</code> is set. Otherwise <code>na_strings()</code> uses the special #N/A value within the workbook.
<code>na.strings</code>	[Deprecated] Use <code>na</code> instead.
<code>overwrite</code>	If FALSE, will not overwrite when <code>file</code> already exists.
<code>call</code>	The execution environment of a currently running function, e.g. <code>caller_env()</code> . The function will be mentioned in error messages as the source of the error. See the <code>call</code> argument of <code>abort()</code> for more information.

Value

Invisibly returns `x` unmodified. The workbook built from `x` is saved to `file` as a side effect.

Examples

```
withr::with_tempdir({
  # Write data frame to XLSX file
  write_xlsx_ext(mtcars, "mtcars.xlsx")

  # Write data frame to XLSX file with workbook title
  write_xlsx_ext(mtcars, title = "mtcars data")

  # Write list of data frames to XLSX file with named sheets
  write_xlsx_ext(
    list(mtcars = mtcars, anscombe = anscombe),
    "datasets-list.xlsx"
  )

  # List output files
  fs::dir_ls()
})
```

Index

* **utils**
 prep_wb_data, 9

abort(), 4, 9, 16, 20, 29
as_data_validation_value, 2
as_data_validation_value(), 2
as_wb, 3
as_wb(), 3, 20, 25

base::as.data.frame(), 9, 15, 28

classic_style(), 8, 17
common_dxfs_styles, 5
convert_to, 5
csv_to_wb (*convert_to*), 5
csv_to_wb(), 6
csv_to_xlsx (*convert_to*), 5
csv_to_xlsx(), 6

dplyr::group_split(), 25

fmt_lgl_cols, 7
fmt_lgl_cols(), 7
fmt_marquee_txt, 8, 17
fmt_marquee_txt(), 8, 16

make.names(), 11, 27
map_wb(), 4, 25
marquee::marquee_parse(), 8
mutate(), 7

na_strings(), 4, 16, 29

openxlsx2::create_dxfs_style(), 5
openxlsx2::fmt_txt(), 8
openxlsx2::read_xlsx, 11
openxlsx2::read_xlsx(), 10
openxlsx2::wb_add_data, 15
openxlsx2::wb_add_data(), 3, 14, 16, 28
openxlsx2::wb_add_data_table, 15
openxlsx2::wb_add_data_table(), 3, 16, 28
openxlsx2::wb_add_data_validation(), 2
openxlsx2::wb_add_style(), 18
openxlsx2::wb_add_worksheet, 19
openxlsx2::wb_add_worksheet(), 18
openxlsx2::wb_get_properties(), 4, 20
openxlsx2::wb_save(), 23
openxlsx2::wb_set_col_widths(), 16
openxlsx2::wb_set_properties(), 24
openxlsx2::wb_to_df, 26
openxlsx2::wb_to_df(), 3, 26
openxlsx2::wb_workbook(), 4, 18, 20, 27
openxlsx2::write_xlsx(), 27

prep_wb_data, 9
prep_wb_data(), 9, 14

read_xlsx_ext, 10
read_xlsx_ext(), 10

set_excel_fmt_class, 12
set_excel_fmt_class(), 13
sf::st_as_text(), 4, 9, 15, 29
sf::st_centroid(), 4, 9, 15, 29
summarise(), 7
Sys.time(), 4, 20

tidyselect::eval_rename(), 21, 22

utils::write.csv(), 6

vctrs::vec_recycle(), 13
vec_fmt_lgl, 13
vec_fmt_lgl(), 7

wb_add_data_ext, 3, 14
wb_add_data_ext(), 14, 27
wb_add_marquee_text, 16
wb_add_marquee_text(), 16

- wb_add_styles, 18
- wb_add_styles(), 5, 18
- wb_color(), 19
- wb_new_workbook, 18
- wb_new_workbook(), 18, 28
- wb_page_setup(), 19
- wb_protect_worksheets, 20
- wb_rename_data, 21
- wb_rename_sheets, 22
- wb_rename_sheets_with
 - (wb_rename_sheets), 22
- wb_save_ext, 23
- wb_save_ext(), 23
- wb_set_properties_ext, 24
- wb_set_properties_ext(), 24
- wb_split, 25
- wb_split(), 25
- wb_to_df_list, 25, 26
- wb_to_df_list(), 25, 26
- wbWorkbook, 11, 18, 25, 26
- write_xlsx_ext, 27
- write_xlsx_ext(), 4, 27

- xlsx_to_csv (convert_to), 5
- xlsx_to_csv(), 6