# Package ‘tatoo’

March 26, 2023

**Type** Package

**Title** Combine and Export Data Frames

**Version** 1.1.2

**Maintainer** Stefan Fleck `<stefan.b.fleck@gmail.com>`

**Description** Functions to combine data.frames in ways that require additional effort in
base R, and to add metadata (id, title, ...) that can be used for printing and
xlsx export. The 'Tatoo_report' class is provided as a
convenient helper to write several such tables to a workbook, one table per
worksheet. Tatoo is built on top of 'openxlsx', but intimate knowledge of
that package is not required to use tatoo.

**License** MIT + file LICENSE

**Imports** assertthat, magrittr, data.table, openxlsx (>= 4.0.0),
stringi, colt, crayon, withr

**Suggests** testthat, rprojroot, kableExtra, knitr, rmarkdown

**RoxygenNote** 7.2.3

**Encoding** UTF-8

**VignetteBuilder** knitr

**BugReports** `https://github.com/statistikat/tatoo/issues`

**URL** `https://github.com/statistikat/tatoo`

**NeedsCompilation** no

**Author** Stefan Fleck [aut, cre]

**Repository** CRAN

**Date/Publication** 2023-03-26 09:50:02 UTC

## R topics documented:

`as.data.table.Composite_table`
*Convert a Composite Table to a data.table or data.frame*

## Description

As a `Composite_table` already is a `data.table` this function does very little except stripping all additional attributes and classes, as well as offering you the option to prepend the `multinames` before the column names

## Usage

```
## S3 method for class 'Composite_table'
as.data.table(x, keep.rownames = NULL, ..., multinames = TRUE, sep = ".")

## S3 method for class 'Composite_table'
as.data.frame(
  x,
  row.names = NULL,
  optional = FALSE,
  ...,
  multinames = TRUE,
  sep = "."
)
```

## Arguments

| | |
|---|---|
| `x` | a `Composite_table` |
| `keep.rownames` | ignored |
| `...` | ignored |
| `multinames` | logical. Whether to prepend multinames before the column names |
| `sep` | separator between multinames and individual column names |
| `row.names` | `NULL` or a character vector giving the row names for the data frame. Missing values are not allowed. |
| `optional` | logical. If `TRUE`, setting row names and converting column names (to syntactic names: see `make.names`) is optional. Note that all of R's **base** package `as.data.frame()` methods use `optional` only for column names treatment, basically with the meaning of `data.frame(*, check.names = !optional)`. See also the `make.names` argument of the `matrix` method. |

## Value

a `data.table` or `data.frame`

`as.data.table.Mashed_table`
*Convert a Mashed Table to a data.table or data.frame*

## Description

Convert a Mashed Table to a data.table or data.frame

## Usage

```
## S3 method for class 'Mashed_table'
as.data.table(
  x,
  keep.rownames = NULL,
  ...,
  mash_method = attr(x, "mash_method"),
  insert_blank_row = attr(x, "insert_blank_row"),
  id_vars = attr(x, "id_vars"),
  suffixes = names(x)
)

## S3 method for class 'Mashed_table'
as.data.frame(
  x,
  row.names = NULL,
  optional = FALSE,
  ...,
  mash_method = attr(x, "mash_method"),
  insert_blank_row = attr(x, "insert_blank_row"),
  id_vars = attr(x, "id_vars"),
  suffixes = names(x)
)
```

## Arguments

| | |
|---|---|
| `x` | a Mashed_table |
| `keep.rownames` | ignored |
| `...` | passed on to `as.data.table()` or `as.data.frame()` respectively |
| `mash_method` | either `"row"` or `"col"`. Should the tables be mashed together with alternating rows or with alternating columns? |
| `insert_blank_row` | Only if mashing rows: logical. Whether to insert blank rows between mash-groups. *Warning: this converts all columns to character.* Use with care. |
| `id_vars` | Only if mashing columns: one ore more colnames of the tables to be mashed. If supplied, columns of both input tables are combined with `merge()`, otherwise `cbind()` is used. |

| | |
|---|---|
| `suffixes` | a character vector of length 2 specifying the suffixes to be used for making unique the names of columns. |
| `row.names` | ignored |
| `optional` | logical. If `TRUE`, setting row names and converting column names (to syntactic names: see `make.names`) is optional. Note that all of R's **base** package `as.data.frame()` methods use `optional` only for column names treatment, basically with the meaning of `data.frame(*, check.names = !optional)`. See also the `make.names` argument of the `matrix` method. |

## Value

a data.table or `data.frame`

## `assign_tt_meta` *Assign tt_meta elements*

### Description

Internal function used by the metadata set functions

### Usage

```
assign_tt_meta(x, assignment)
```

### Arguments

| | |
|---|---|
| `x` | a Tatoo_table or data.frame |
| `assignment` | A named list of length one, for example `list(longtitle = value)` |

## `as_Composite_table` *Coerce to Composite Table*

### Description

Converts other R objects to `Composite_tables` by automatically creating multi-column names from the properties of the objects.

## Usage

```
as_Composite_table(x, ...)

## S3 method for class 'Mashed_table'
as_Composite_table(
  x,
  id_vars = attr(x, "id_vars"),
  meta = attr(x, "meta"),
  ...
)

## S3 method for class 'data.frame'
as_Composite_table(x, sep = ".", reverse = FALSE, ...)

is_Composite_table(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | Any R object. |
| `...` | Ignored |
| `id_vars` | If `id_vars` is specified, the tables will be combined using `merge()` on the columns specified in `id_vars`, otherwise the tables will be combined with `cbind()`. |
| `meta` | a TT_meta object. If specified, the resulting `Composite_table` will be wrapped in a Tagged_table. |
| `sep` | a scalar character. Separator in the column names of `x` that separates the column name from the multi-column name. |
| `reverse` | logical. if `FALSE` the part after the last occurrence of `sep` will be used as multi-name, if `TRUE` the part before will be used. |

## Value

`as_Composte_table()` returns a `Composite_table`

`is_Composite_table` returns `TRUE` if its argument is a `Composite_table` and `FALSE` otherwise.

## Examples

```
mash_table(
  head = head(cars),
  tail = tail(cars),
  mash_method = 'col'
)


as_Composite_table(data.frame(
  apple.fruit = 1,
  kiwi.fruit = 2,
  dog.animal = 1,
  black.cat.animal = 2,
  parrot.animal = 3
))
```

## as_latex *Convert a Table to Latex Code*

### Description

`as_latex()` converts an R Object (currently `Tatoo_table`s and `data.frames`) to latex code.

`save_pdf()` is a wrapper around `as_latex()` for directly saving an R object to '`.pdf`'.

`view_pdf()` is another wrapper for directly viewing an R Object's pdf representation on a pdf viewer (powered by `open_file()`).

### Usage

```
as_latex(x, ..., kable_options = default_kable_options())

save_pdf(
  x,
  outfile,
  ...,
  overwrite = FALSE,
  papersize = "a4paper",
  orientation = "portrait",
  keep_source = FALSE,
  template = system.file("templates", "save_tex.Rmd", package = "tatoo")
)

view_pdf(x, ...)
```

### Arguments

| | |
|---|---|
| `x` | a `Tatoo_table`, `data.frame` or a list of `data.frames` |
| `...` | passed on to methods |
| `kable_options` | `list`. Options passed on to `knitr::kable()`. See `default_kable_options()` for details. |
| `outfile` | `character` scalar. Path to the output file |
| `overwrite` | If `TRUE`, overwrite any existing file. |
| `papersize` | `character` scalar. Passed on to the latex command `\\geometry` from the 'geometry' package. Valid values are: `a0paper, a1paper, a2paper, a3paper, a4paper, a5paper, a6pa` |
| `orientation` | `character` scalar. Passed on to the latex command `\\geometry` from the 'geometry' package. Valid values are: `portrait`, `landscape` |
| `keep_source` | When saving a '`pdf`', also put the Latex source in the same directory. |
| `template` | Latex template for the desired output. Use the template file supplied in this package if you want to create your own. |

### Details

as_latex() and co. are designed to produce nice looking output with a minimum of user input required. This is useful if you want a quick preview or printout of a table. If you need customized Latex the output, you should take a look at the packages kableExtra::kableExtra, **xtable**, or **huxtable**.

### Value

as_latex()returns a character scalar of Latex code

save_pdf() returns a the path to the saved file as character scalar.

view_pdf() returns NULL (invisibly)

### Latex Packages

as_latex requires that the following Latex packages are installed on your system:

```
\usepackage{booktabs}
\usepackage{longtable}
\usepackage{threeparttablex}
```

### See Also

Other as_latex methods: as_latex.Composite_table(), as_latex.Mashed_table(), as_latex.Tagged_table(), as_latex.Tatoo_report(), as_latex.data.frame()

### Examples

```
  as_latex(iris)

## Not run:
  view_pdf(iris)  # Not supported on all systems

## End(Not run)
```

---

## as_latex.Composite_table

*Convert a Composite Table to Latex Code*

### Description

Convert a Composite Table to Latex Code

### Usage

```
## S3 method for class 'Composite_table'
as_latex(x, id_vars = id_vars(x), ..., kable_options = default_kable_options())
```

## Arguments

| | |
|---|---|
| x | a Tatoo_table, data.frame or a list of data.frames |
| id_vars | If id_vars is specified, the tables will be combined using merge() on the columns specified in id_vars, otherwise the tables will be combined with cbind(). |
| ... | comp_table() only: individual data.frames. A name can be provided for each data.frame that will be used by print() and as_workbook() to create multi-table headings. |
| kable_options | list. Options passed on to knitr::kable(). See default_kable_options() for details. |

## Value

as_latex()returns a character scalar of Latex code

save_pdf() returns a the path to the saved file as character scalar.

view_pdf() returns NULL (invisibly)

## See Also

Other as_latex methods: as_latex.Mashed_table(), as_latex.Tagged_table(), as_latex.Tatoo_report(), as_latex.data.frame(), as_latex()

---

as_latex.data.frame *Convert a Data Frame to Latex Code*

---

## Description

Convert a Data Frame to Latex Code

## Usage

```
## S3 method for class 'data.frame'
as_latex(x, ..., kable_options = default_kable_options())
```

## Arguments

| | |
|---|---|
| x | a Tatoo_table, data.frame or a list of data.frames |
| ... | passed on to methods |
| kable_options | list. Options passed on to knitr::kable(). See default_kable_options() for details. |

## Value

as_latex()returns a character scalar of Latex code

save_pdf() returns a the path to the saved file as character scalar.

view_pdf() returns NULL (invisibly)

## See Also

Other as_latex methods: `as_latex.Composite_table()`, `as_latex.Mashed_table()`, `as_latex.Tagged_table()`, `as_latex.Tatoo_report()`, `as_latex()`

---

`as_latex.Mashed_table` *Convert a Mashed Table to Latex Code*

---

## Description

Convert a Mashed Table to Latex Code

## Usage

```
## S3 method for class 'Mashed_table'
as_latex(
  x,
  mash_method = attr(x, "mash_method"),
  id_vars = attr(x, "id_vars"),
  insert_blank_row = attr(x, "insert_blank_row"),
  sep_height = attr(x, "sep_height"),
  ...,
  kable_options = default_kable_options()
)
```

## Arguments

| | |
|---|---|
| `x` | a `Tatoo_table`, `data.frame` or a list of `data.frames` |
| `mash_method` | either `"row"` or `"col"`. Should the tables be mashed together with alternating rows or with alternating columns? |
| `id_vars` | Only if mashing columns: one ore more colnames of the tables to be mashed. If supplied, columns of both input tables are combined with `merge()`, otherwise `cbind()` is used. |
| `insert_blank_row` | Only if mashing rows: logical. Whether to insert blank rows between mash-groups. *Warning: this converts all columns to character.* Use with care. |
| `sep_height` | Only has an effect when exporting to `xlsx`. if `insert_blank_row` == `TRUE`, height of the inserted row, else height of the top row of each mash-group. |
| `...` | `mash_table()` only: `data.frames` with the same row and column count. Elements of `(...)` can be named, but the name must differ from the argument names of this function. |
| `kable_options` | `list`. Options passed on to `knitr::kable()`. See `default_kable_options()` for details. |

## Value

`as_latex()`returns a `character` scalar of Latex code

`save_pdf()` returns a the path to the saved file as `character` scalar.

`view_pdf()` returns `NULL` (invisibly)

## See Also

Other as_latex methods: `as_latex.Composite_table()`, `as_latex.Tagged_table()`, `as_latex.Tatoo_report()`, `as_latex.data.frame()`, `as_latex()`

---

## `as_latex.Tagged_table` *Convert a Tagged Table to Latex Code*

---

### Description

Convert a Tagged Table to Latex Code

### Usage

```
## S3 method for class 'Tagged_table'
as_latex(x, ..., kable_options = default_kable_options())
```

### Arguments

| | |
|---|---|
| `x` | a `Tatoo_table`, `data.frame` or a list of `data.frames` |
| `...` | passed on to methods |
| `kable_options` | `list`. Options passed on to `knitr::kable()`. See `default_kable_options()` for details. |

### Value

`as_latex()`returns a `character` scalar of Latex code

`save_pdf()` returns a the path to the saved file as `character` scalar.

`view_pdf()` returns `NULL` (invisibly)

### See Also

Other as_latex methods: `as_latex.Composite_table()`, `as_latex.Mashed_table()`, `as_latex.Tatoo_report()`, `as_latex.data.frame()`, `as_latex()`

## `as_latex.Tatoo_report` *Convert a Tatoo Report to Latex Code*

### Description

Convert a Tatoo Report to Latex Code

### Usage

```
## S3 method for class 'Tatoo_report'
as_latex(x, ...)
```

### Arguments

| | |
|---|---|
| `x` | a `Tatoo_table`, `data.frame` or a list of `data.frames` |
| `...` | for compile_table: `individual Tatoo_table` or data.frame‘ objects |

### Value

`as_latex()`returns a `character` scalar of Latex code

`save_pdf()` returns a the path to the saved file as `character` scalar.

`view_pdf()` returns `NULL` (invisibly)

### See Also

Other as_latex methods: `as_latex.Composite_table()`, `as_latex.Mashed_table()`, `as_latex.Tagged_table()`, `as_latex.data.frame()`, `as_latex()`

## `as_lines` *Create a line-by-line text representation of an R object*

### Description

Creates a line-by-line representation of an R object (usually a `Tatoo_table`). This is the function powers all `Tatoo_table` print methods.

### Usage

```
as_lines(x, color = TRUE, ...)

## S3 method for class 'data.frame'
as_lines(x, color = TRUE, ...)

## S3 method for class 'Tagged_table'
as_lines(x, color = TRUE, ...)
```

```
## S3 method for class 'Mashed_table'
as_lines(
  x,
  color = TRUE,
  mash_method = attr(x, "mash_method"),
  insert_blank_row = attr(x, "insert_blank_row"),
  id_vars = attr(x, "id_vars"),
  ...
)

## S3 method for class 'Stacked_table'
as_lines(x, color = TRUE, ...)

## S3 method for class 'Composite_table'
as_lines(x, color = TRUE, ...)

## S3 method for class 'Tatoo_report'
as_lines(x, color = TRUE, ...)

## S3 method for class 'TT_meta'
as_lines(x, color = TRUE, ...)
```

### Arguments

| | |
|---|---|
| `x` | Any R object. |
| `color` | Use colors (via **colt**) |
| `...` | passed on methods. |
| `mash_method` | either `"row"` or `"col"`. Should the tables be mashed together with alternating rows or with alternating columns? |
| `insert_blank_row` | Only if mashing rows: logical. Whether to insert blank rows between mash-groups. *Warning: this converts all columns to character.* Use with care. |
| `id_vars` | Only if mashing columns: one ore more colnames of the tables to be mashed. If supplied, columns of both input tables are combined with `merge()`, otherwise `cbind()` is used. |

### Value

A character vector (one element per line).

---

## `as_Mashed_table` *Coerce to Mashed Table*

---

### Description

Coerce to Mashed Table

### Usage

```
as_Mashed_table(x, ...)

is_Mashed_table(x, ...)
```

### Arguments

| | |
|---|---|
| `x` | Any R object. |
| `...` | `mash_table()` only: `data.frames` with the same row and column count. Elements of `(...)` can be named, but the name must differ from the argument names of this function. |

### Value

`as_Mashed_table()` returns a Mashed_table

`is_Mashed_table` returns `TRUE` if its argument is a `Mashed_table` and `FALSE` otherwise.

---

## `as_multinames` *Create Composite Table multinames from a character vector*

---

### Description

Create Composite Table multinames from a character vector

### Usage

```
as_multinames(x)
```

### Arguments

| | |
|---|---|
| `x` | a character vector of equal length as the data.frame for which it the multinames should be created. |

### Value

a named integer vector that can be used as multinames attribute for a Composite_table

### Examples

```
dat <- data.frame(
  apple = 1,
  banana = 2,
  dog = 1,
  cat = 2,
  parrot = 3
)
```

```
multinames(dat) <- as_multinames(
  c('fruit', 'fruit', 'animal', 'animal', 'animal')
)

multinames(dat)
```

---

`as_workbook` *Convert a Tatoo Table Object to an Excel Workbook*

---

## Description

`as_workbook()` converts `Tatoo_table` or `Tatoo_report` objects directly to openxlsx `Workbook` objects. For information about additional parameters please refer to the documentation of `write_worksheet()`, for which `as_workbook()` is just a wrapper. Additional possible function arguments way vary depending on which `Tatoo_table` you want to export.

`save_xlsx()` is a wrapper for saving a `Tatoo_table` directly to an '`xlsx`' file.

`view_xlsx()` is another wrapper for viewing a `Tatoo_table`"s '`xlsx`' representation in your favorite spreadsheet program (powered by `openxlsx::openXL()`).

## Usage

```
as_workbook(x, ...)

## Default S3 method:
as_workbook(x, sheet = 1L, ...)

## S3 method for class 'Tatoo_report'
as_workbook(x, ...)

save_xlsx(x, outfile, overwrite = FALSE, ...)

view_xlsx(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | A `Tatoo_table` or `Tatoo_report` |
| `...` | Additional arguments passed on to `write_worksheet()` |
| `sheet` | The worksheet to write to. Can be the worksheet index or name. |
| `outfile` | Path to the output file |
| `overwrite` | If `TRUE`, overwrite any existing file. |

## Value

`as_workbook()` returns an openxlsx `Workbook` object.

`save_xlsx()` returns the path to the saved '`.xlsx`' (invisibly).

`view_xlsx()` opens an external program and returns `NULL` (invisibly).

## See Also

Other xlsx exporters: `write_worksheet()`

## Examples

```
## Not run:
dat <- data.frame(
  Species = c("setosa", "versicolor", "virginica"),
  length = c(5.01, 5.94, 6.59),
  width = c(3.43, 2.77, 2.97)
)

# Assign metadata to convert dat to a Tagged_table

title(dat) <- "Iris excerpt"
footer(dat) <-  "An example based on the iris dataset"


# Convert to Workbook or save als xlsx

wb <- as_workbook(dat)
save_xlsx(dat, tempfile(fileext = ".xlsx"), overwrite = TRUE)

## End(Not run)
```

---

`compile_report` *Compile Tables Into a Report*

---

## Description

Compiles tables into a `Tatoo_report`. A `Tatoo_report` is just a simple list object, but with special `print`, `as_workbook`, and `save_xlsx` methods. This makes it easy to save an arbitrary number of tables to a single Excel workbook.

## Usage

```
compile_report(...)

compile_report_list(dat)
```

## Arguments

| | |
|---|---|
| `...` | for compile_table: `individual Tatoo_table` or data.frame‘ objects |
| `dat` | for `compile_table_list`: A list of containing either `Tatoo_table` or `data.frame` objects. |

## Value

A `Tatoo_report`: A list whose elements are either `data.frames` or `Tatoo_tables`

---

`comp_table` *Compose Tables*

---

## Description

`comp_table()` is a drop in replacement for `base::cbind()` that supports multi-column headings.#'

## Usage

```
comp_table(..., id_vars = NULL, meta = NULL)

comp_table_list(tables, id_vars = NULL, meta = NULL)
```

## Arguments

| | |
|---|---|
| `...` | `comp_table()` only: individual `data.frames`. A name can be provided for each `data.frame` that will be used by `print()` and `as_workbook()` to create multi-table headings. |
| `id_vars` | If `id_vars` is specified, the tables will be combined using `merge()` on the columns specified in `id_vars`, otherwise the tables will be combined with `cbind()`. |
| `meta` | a TT_meta object. If specified, the resulting `Composite_table` will be wrapped in a Tagged_table. |
| `tables` | `comp_table_list` only: A named list of data.frames with the same number of rows |

## Value

A `Composite_table`.

## See Also

Attribute setter: multinames<-

Other Tatoo tables: `mash_table()`, `stack_table()`, `tag_table()`, `tatoo_table()`

## Examples

```
df_mean <- data.frame(
  Species = c("setosa", "versicolor", "virginica"),
  length = c(5.01, 5.94, 6.59),
  width = c(3.43, 2.77, 2.97)
)

df_sd <- data.frame(
  Species = c("setosa", "versicolor", "virginica"),
  length = c(0.35, 0.52, 0.64),
  width = c(0.38, 0.31, 0.32)
)

comp_table(mean = df_mean, sd = df_sd)

# ...........mean............     ............sd.............
# 1    Species   length   width        Species   length   width
# 2     setosa     5.01    3.43         setosa     0.35    0.38
# 3 versicolor     5.94    2.77     versicolor     0.52    0.31
# 4  virginica     6.59    2.97      virginica     0.64    0.32


comp_table(mean = df_mean, sd = df_sd, id_vars = 'Species')

# ..........     .....mean.....     ......sd......
# 1    Species     length   width     length   width
# 2     setosa       5.01    3.43       0.35    0.38
# 3 versicolor       5.94    2.77       0.52    0.31
# 4  virginica       6.59    2.97       0.64    0.32
```

---

`default_kable_options` *Default Kable options for as_latex and co*

---

## Description

`default_kable_options()` returns a list of the default options that are required for `as_latex()` to work correctly. Those defaults should not be modified, but you can pass additional `knitr::kable()` options to `as_latex()` to modify the output a bit.

## Usage

```
default_kable_options(...)
```

## Arguments

| | |
|---|---|
| `...` | additional arguments added to the options list |

## Examples

```

default_kable_options

as_latex(iris, kable_options = default_kable_options(digits = 0))
```

## `df_typecast_all` *Typecast all columns of a data.frame of a specific type*

### Description

Bulk convert columns of a data.frame that share a certain class to a different class. Use with care, will introduce NAs for some conversion attempts

### Usage

```
df_typecast_all(dat, from = "factor", to = "character")
```

### Arguments

| | |
|---|---|
| `dat` | a data.frame |
| `from` | column type to cast |
| `to` | target column type |

### Value

a data frame with all columns of class from converted to class to

## `flip_names` *Flip names and multinames of a Composite Table*

### Description

The column names of the resulting `Composite_table` will be sorted lexically

### Usage

```
flip_names(dat, id_vars)
```

### Arguments

| | |
|---|---|
| `dat` | A `Composite_table` |
| `id_vars` | a character vector of column names of `dat`. The selected columns will not be sorted lexically but kept to the left. If the columns have a multiname associated with them, they must be supplied in the format `column_name.multiname`. |

### Value

a Composite_table

### Examples

```
dat <- comp_table(
  cars1 = head(cars),
  cars2 = tail(cars),
  data.frame(id = LETTERS[1:6])
)

flip_names(dat)
flip_names(dat, id_vars = "id")
flip_names(dat, id_vars = c("id", "speed.cars1"))
```

## is_any_class *Check if any of the classes of the object match a certain string*

### Description

Check if any of the classes of the object match a certain string

### Usage

```
is_any_class(dat, choices)
```

### Arguments

| | |
|---|---|
| dat | the object |
| choices | the class to be checked for |

### Value

True if any of the object classes are the desired class

## is_class *Check if object is of a certain class*

### Description

These functions are designed to be used in combination with the assertthat package

### Usage

```
is_class(dat, class)

assert_class(dat, class)

dat %assert_class% class
```

### Arguments

| | |
|---|---|
| `dat` | any R object |
| `class` | the class to be checked for |

### Details

'is_class returns()' 'TRUE'/'FALSE'. It comes with a on_failure function and is designed to be used in conjunction with the assertthat package. 'assert_class()' and its infix version

### Value

'is_class()' returns 'TRUE'/'FALSE', 'assert_class()' returns 'TRUE' or fails with an error message.

---

## `is_col_classes` *Check for column classes*

---

### Description

Compares the column classes of a data.frame with

### Usage

```
is_col_classes(dat, classes, method = "identical")
```

### Arguments

| | |
|---|---|
| `dat` | a data.frame or list |
| `classes` | a list of column classes. Its names must match the names of dat exactly (see example) |
| `method` | if `all`, ensure that all columns named in `classes` are present in `dat`, if `any`, ensure that any of the columns named in `classes` are present in `dat`, if `identical`, ensure that the names of dat and classes are identical |

## `is_Stacked_table` *Test If Object is a Stacked_table*

### Description

Test If Object is a Stacked_table

### Usage

```
is_Stacked_table(x)
```

### Arguments

| | |
|---|---|
| `x` | Any R object. |

### Value

`is_Stacked_table()` returns `TRUE` if its argument is a `Stacked_table` and `FALSE` otherwise.

## `is_Tagged_table` *Test If Object is a Tagged_table*

### Description

Test If Object is a Tagged_table

### Usage

```
is_Tagged_table(x)
```

### Arguments

| | |
|---|---|
| `x` | Any R object. |

## `is_Tatoo_report` *Test if Object is a Tatoo_report*

### Description

Test if Object is a Tatoo_report

### Usage

```
is_Tatoo_report(x)
```

### Arguments

| | |
|---|---|
| `x` | Any R object. |

### Value

`is_Tatoo_report()` returns `TRUE` if its argument is a `Tatoo_report` and `FALSE` otherwise.

## `is_Tatoo_table` *Test if objects is a Tatoo_table*

### Description

Test if objects is a Tatoo_table

### Usage

```
is_Tatoo_table(x)
```

### Arguments

| | |
|---|---|
| `x` | Any R object. |

### Value

`is_Tatoo_table` returns `TRUE` if its argument is a `Tatoo_table` and `FALSE` otherwise.

## `mash_method<-` *Set mash attributes of a Mashed Table*

### Description

Set mash attributes of a Mashed Table

### Usage

```
mash_method(x) <- value

insert_blank_row(x) <- value

sep_height(x) <- value

id_vars(x) <- value
```

### Arguments

| | |
|---|---|
| `x` | a `Mashed_table` |
| `value` | a value that is legal for the individual attribute, as described in Mashed_table |

### See Also

Mashed_table

## `mash_table` *Mash Tables*

### Description

`mash_tables()` makes it easy to put together multidimensional tables from `data.frames` with the same number of rows and columns. You can mash tables together with either alternating rows or columns.

### Usage

```
mash_table(
  ...,
  mash_method = "row",
  id_vars = NULL,
  insert_blank_row = FALSE,
  sep_height = 24,
  meta = NULL,
  rem_ext = NULL
)

mash_table_list(
  tables,
  mash_method = "row",
  id_vars = NULL,
  insert_blank_row = FALSE,
  sep_height = 24,
  meta = NULL,
  rem_ext = NULL
)
```

## Arguments

| | |
|---|---|
| `...` | `mash_table()` only: `data.frames` with the same row and column count. Elements of `(...)` can be named, but the name must differ from the argument names of this function. |
| `mash_method` | either `"row"` or `"col"`. Should the tables be mashed together with alternating rows or with alternating columns? |
| `id_vars` | Only if mashing columns: one ore more colnames of the tables to be mashed. If supplied, columns of both input tables are combined with `merge()`, otherwise `cbind()` is used. |
| `insert_blank_row` | Only if mashing rows: logical. Whether to insert blank rows between mash-groups. *Warning: this converts all columns to character.* Use with care. |
| `sep_height` | Only has an effect when exporting to `xlsx`. if `insert_blank_row == TRUE`, height of the inserted row, else height of the top row of each mash-group. |
| `meta` | A TT_meta object. if supplied, output will also be a Tagged_table. |
| `rem_ext` | `character`. For `mash_table` to work, the column names of all elements of `dat` must be identical. Sometimes you will have the situation that column names are identical except for a suffix, such as `length` and `lenght.sd`. The `rem_ext` option can be used to remove such suffixes. |
| `tables` | `mash_table_list()` only: a `list` of `data.frames` as described for `(...)` |

## Value

a `Mashed_table`: a `list` of `data.tables` with additional `mash_method`, `insert_blank_row` and `sep_height` attributes, that influence how the table looks when it is printed or exported.

## See Also

Attribute setters: mash_method<-

Other Tatoo tables: `comp_table()`, `stack_table()`, `tag_table()`, `tatoo_table()`

## Examples

```
df_mean <- data.frame(
  Species = c("setosa", "versicolor", "virginica"),
  length = c(5.01, 5.94, 6.59),
  width = c(3.43, 2.77, 2.97)
)

df_sd <- data.frame(
  Species = c("setosa", "versicolor", "virginica"),
  length = c(0.35, 0.52, 0.64),
  width = c(0.38, 0.31, 0.32)
)


# Mash by row

mash_table(df_mean, df_sd)

#       Species length width
# 1:     setosa   5.01  3.43
# 2:     setosa   0.35  0.38
# 3: versicolor   5.94  2.77
# 4: versicolor   0.52  0.31
# 5:  virginica   6.59  2.97
# 6:  virginica   0.64  0.32


# Mash by column

mash_table(
  df_mean, df_sd,
  mash_method = 'col',
  id_vars = 'Species'
)

#       Species    Species length length width width
# 1:     setosa     setosa   5.01   0.35  3.43  0.38
# 2: versicolor versicolor   5.94   0.52  2.77  0.31
# 3:  virginica  virginica   6.59   0.64  2.97  0.32


# Use the id_vars argument to prevent undesired dpulicated columns,
# and name the input data.frames to get multi-col headings.

mash_table(
  mean = df_mean, sd = df_sd,
  mash_method = 'col',
  id_vars = 'Species'
)

#    ..........     ..length...     ...width...
# 1    Species     mean     sd      mean     sd
# 2     setosa     5.01   0.35      3.43   0.38
# 3 versicolor     5.94   0.52      2.77   0.31
# 4  virginica     6.59   0.64      2.97   0.32
```

## `meta<-` *Set Tagged Table metadata*

### Description

Convenience functions to modify `Tagged_table` metadata. If `x` is not a `Tagged_table` already, it will be converted to one.

### Usage

```
meta(x) <- value

meta(x)

table_id(x) <- value

table_id(x)

title(x) <- value

longtitle(x) <- value

subtitle(x) <- value

footer(x) <- value
```

### Arguments

| | |
|---|---|
| `x` | a `Tagged_table` or any R object that can be converted to one |
| `value` | value to assign. |

### See Also

Tagged_table, tt_meta

## multinames<- *Set the multinames attribute of a Composite_table*

### Description

Set the multinames attribute of a Composite_table

### Usage

```
multinames(x) <- value

multinames(x)
```

### Arguments

| | |
|---|---|
| `x` | a `Composite_table` or `data.frame` |
| `value` | a named vector of ascending integers. The name is the multi-column heading, the integer value is the last column that this heading applies to |

### See Also

Composite_table, `as_multinames()`

### Examples

```

df_mean <- data.frame(
  Species = c("setosa", "versicolor", "virginica"),
  length = c(5.01, 5.94, 6.59),
  width = c(3.43, 2.77, 2.97)
)

multinames(df_mean) = c("species" = 1, measures = 3)

# .species..     ...measures...
# 1    Species     length   width
# 2     setosa       5.01    3.43
# 3 versicolor       5.94    2.77
# 4  virginica       6.59    2.97
```

## `multinames_to_colspans` *Convert multinames to colspans*

### Description

Convert multinames to colspans

### Usage

```
multinames_to_colspans(x)
```

### Arguments

| | |
|---|---|
| `x` | a Composite_table multinames attribute. |

### Value

A named character vector of colspans (for `kableExtra::add_header_above()`)

## `open_file` *Open a file*

### Description

Open a file with the default associated program. Might behave differently depending on the operating system.

### Usage

```
open_file(x)
```

### Arguments

| | |
|---|---|
| `x` | `character` scalar. Path to the file to open. |

### Value

`NULL` (invisibly)

## `print.Composite_table` *Printing Composite Tables*

### Description

Printing Composite Tables

### Usage

```
## S3 method for class 'Composite_table'
print(x, right = FALSE, ...)
```

### Arguments

| | |
|---|---|
| `x` | a `Composite_table` |
| `right` | Logical. Should strings be right aligned? The default is left-alignment (the opposite of the standard `print.data.frame()`). |
| `...` | passed on to `print` |

### Value

`x` (invisibly)

## `print.Mashed_table` *Printing Mashed Tables*

### Description

Printing Mashed Tables

### Usage

```
## S3 method for class 'Mashed_table'
print(
  x,
  mash_method = attr(x, "mash_method"),
  insert_blank_row = attr(x, "insert_blank_row"),
  id_vars = attr(x, "id_vars"),
  ...
)
```

### Arguments

| | |
|---|---|
| `x` | a Mashed_table |
| `mash_method` | either `"row"` or `"col"`. Should the tables be mashed together with alternating rows or with alternating columns? |
| `insert_blank_row` | Only if mashing rows: logical. Whether to insert blank rows between mash-groups. *Warning: this converts all columns to character.* Use with care. |
| `id_vars` | Only if mashing columns: one ore more colnames of the tables to be mashed. If supplied, columns of both input tables are combined with `merge()`, otherwise `cbind()` is used. |
| `...` | passed on to `print()` |

### Value

x (invisibly)

---

## `print.Stacked_table` *Printing Stacked Tables*

---

### Description

Printing Stacked Tables

### Usage

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

### Arguments

| | |
|---|---|
| `x` | A `Stacked_table` |
| `...` | passed on to `print()` |

### Value

x (invisibly)

## `print.Tagged_table` *Printing Tagged Tables*

### Description

Printing Tagged Tables

### Usage

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

### Arguments

| | |
|---|---|
| `x` | a Tagged_table |
| `...` | passed on to `print()` |

### Value

`x` (invisibly)

## `print.Tatoo_report` *Printing Tatoo Reports*

### Description

Printing Tatoo Reports

### Usage

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

### Arguments

| | |
|---|---|
| `x` | A `Tatoo_report` |
| `...` | passed on to `print` |

### Value

`x` (invisibly)

## `print.TT_meta` *Printing Tagged Table Metadata*

### Description

Printing Tagged Table Metadata

### Usage

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

### Arguments

| | |
|---|---|
| `x` | A `TT_meta` object |
| `...` | Ignored |

### Value

`x` (invisibly)

## `regions` *Get Named Regions of an Excel Sheet as Data.Table*

### Description

Get Named Regions of an Excel Sheet as Data.Table

### Usage

```
regions(x)
```

### Arguments

| | |
|---|---|
| `x` | An openxlsx workbook or a `character` vector with attributes `position` and `sheet` as returned by `openxlsx::getNamedRegions()` |

### Value

A `data.table`

## rmash *Mash R objects by Rows or Columns*

### Description

rmash() and cmash() are convenience function to mash data.frames together with a single command. They behave similar to cbind() and rbind(), just that the result will have have alternating rows/columns.

### Usage

```
rmash(..., rem_ext = NULL, insert_blank_row = FALSE, meta = NULL)

cmash(
  ...,
  rem_ext = NULL,
  id_vars = NULL,
  suffixes = names(list(...)),
  meta = NULL
)
```

### Arguments

| | |
|---|---|
| ... | either several data.frames, data.tables or a single Mashed_table. All data.frames must have the same number of columns. |
| rem_ext | character. For mash_table to work, the column names of all elements of dat must be identical. Sometimes you will have the situation that column names are identical except for a suffix, such as length and lenght.sd. The rem_ext option can be used to remove such suffixes. |
| insert_blank_row | Only if mashing rows: logical. Whether to insert blank rows between mash-groups. *Warning: this converts all columns to character.* Use with care. |
| meta | A TT_meta object. if supplied, output will also be a Tagged_table. |
| id_vars | Only if mashing columns: one ore more colnames of the tables to be mashed. If supplied, columns of both input tables are combined with merge(), otherwise cbind() is used. |
| suffixes | a character vector of length 2 specifying the suffixes to be used for making unique the names of columns. |

### Value

A data.table if any element of (...) is a data.table or Tatoo_table, or if meta is supplied; else a data.frame.

### See Also

Mashed_table

## Examples

```
dat1 <- data.frame(
  x = 1:3,
  y = 4:6
)

dat2 <- data.frame(
  x = letters[1:3],
  y = letters[4:6]
)

rmash(dat1, dat2)

#    x y
# 1: 1 4
# 2: a d
# 3: 2 5
# 4: b e
# 5: 3 6
# 6: c f

cmash(dat1, dat2)

#    x x y y
# 1: 1 a 4 d
# 2: 2 b 5 e
# 3: 3 c 6 f
```

---

`sanitize_excel_sheet_names` *Sanitize excel sheet names*

---

## Description

Convert a vector to valid excel sheet names by:

- trimming names down to 31 characters,
- ensuring each element of the vector is unique, and
- removing the illegal characters `\ / * [ ] : ?`

[ ]: R:%20

## Usage

```
sanitize_excel_sheet_names(x, replace = "_")
```

### Arguments

| | |
|---|---|
| x | a vector (or anything that can be coerced to one via `as.character()`). |
| replace | a scalar character to replace illegal characters with |

### Value

a character vector of valid excel sheet names

### Examples

```
sanitize_excel_sheet_names(
  c("a very: long : vector? containing some illegal characters",
    "a very: long : vector? containing some illegal characters")
)

  # [1] "a very_ long  vector_ containi0" "a very_ long  vector_ containi1"
```

## `spacing<-` *Set the spacing of a Stacked_table*

### Description

Set the number of lineskips between the tables when exporting to xlsx.

### Usage

```
spacing(x) <- value
```

### Arguments

| | |
|---|---|
| x | a `Stacked_table` |
| value | a scalar integer |

### See Also

Stacked_table

## stack_table *Stack Tables*

### Description

Stack tables on top of each other. This can be used to print several tables on one Excel sheet with `as_workbook()` or `save_xlsx()`.

### Usage

```
stack_table(..., spacing = 2L, meta = NULL)

stack_table_list(tables, spacing = 2L, meta = NULL)
```

### Arguments

| | |
|---|---|
| `...` | `stack_table()` only: Any number other `Tatoo_table` objects, or anything that can be coerced to a `data.frame`. |
| `spacing` | Number of lineskips between the tables when exporting to xlsx |
| `meta` | a `tt_meta` object (optional) |
| `tables` | `stack_table_list()` only: Same as `(...)` for `stack_table`, just that a list can be supplied instead of individual arguments. |

### Value

A Stacked_table: a list of `Tatoo_tables` with additional `spacing` attribute that controls the default spacing between the tables when it is exported.

### See Also

Attribute setter: spacing<-

Other Tatoo tables: `comp_table()`, `mash_table()`, `tag_table()`, `tatoo_table()`

### Examples

```

df1 <- iris[1:5, 3:5]
df2 <- iris[100:105, 3:5]

stack_table(df1, df2)

# ````````````````````````````````````````````
# `      Petal.Length Petal.Width Species
# `   1:          1.4         0.2  setosa
# `   2:          1.4         0.2  setosa
# `   3:          1.3         0.2  setosa
# `   4:          1.5         0.2  setosa
```

```
# `   5:          1.4         0.2  setosa
# `   --------------------------------------
# `      Petal.Length Petal.Width    Species
# `   1:          4.1         1.3 versicolor
# `   2:          6.0         2.5  virginica
# `   3:          5.1         1.9  virginica
# `   4:          5.9         2.1  virginica
# `   5:          5.6         1.8  virginica
# `   6:          5.8         2.2  virginica
# `
# ``````````````````````````````````````````
```

## `str_nobreak` *Remove linebreaks and multiple spaces from string*

### Description

Remove linebreaks and multiple spaces from string

### Usage

```
str_nobreak(x)
```

### Arguments

| | |
|---|---|
| `x` | a character vector. |

### Value

a character vector without linebreaks

## `tag_table` *Tag Tables*

### Description

Add metadata/captioning (like `table_id`, `title`, `footer`) to a `Tatoo_table` or `data.frame`. This metadata will be used by `print()` methods and export functions such as `as_workbook()` or `save_xlsx()`.

### Usage

```
tag_table(dat, meta)
```

## Arguments

| | |
|---|---|
| `dat` | A `Tatto_table` object or anything that can be coerced to a `data.table`. |
| `meta` | a `tt_meta` object. Metadata can also be set and modified using setters (see `meta()`) |

## Value

a `Tagged_table`: a `Tatoo_table` with an additional `meta` attribute

## See Also

Attribute setters: `meta<-()`

Tagged Table Metadata: `tt_meta()`

Other Tatoo tables: `comp_table()`, `mash_table()`, `stack_table()`, `tatoo_table()`

## Examples

```
dat <- data.frame(
  name  = c("hans", "franz", "dolores"),
  grade = c(1, 3, 2)
)

table_metadata <- tt_meta(
  table_id = "Tab1",
  title = "Grades",
  longtitle = "grades of the final examination"
)

# Metadata can be assign in a formal way or via set functions
dat <- tag_table(dat,  meta = table_metadata)
meta(dat) <- table_metadata

# Table metadata is stored as an attribute, and cann be acces thus. It can
# also be modified via convenient set functions
attr(dat, 'meta')$title
meta(dat)$title
longtitle(dat) <- "Grades of the final examination"

# [1] "Grades"

print(dat)

# Tab1: Grades - Grades of the final examination
#
# name grade
# 1:    hans     1
# 2:   franz     3
# 3: dolores     2
```

## tatoo *tatoo: Combine and Export Data Frames*

### Description

Functions to combine data.frames in ways that require additional effort in base R, and to add metadata (id, title, ...) that can be used for printing and xlsx export. The 'Tatoo_report' class is provided as a convenient helper to write several such tables to a workbook, one table per worksheet. Tatoo is built on top of 'openxlsx', but intimate knowledge of that package is not required to use tatoo.

### Functions

- `tag_table()`: add captioning (title, footer, ...) to a table
- `comp_table()`: like `cbind()` or `merge()`, but retain multi-column headings
- `mash_table()`: combine data.frames so that their rows or columns alternate. Mash tables are stored as lists that can be converted to data.tables, or you can use `rmash()` and `cmash()` to create `data.frames` directly.
- `stack_table()`: create a list of tables that can be exported to xlsx, all tables on the same worksheet on top of each others
- `compile_report()`: create a list of tables that can be exported to xlsx, one table per worksheet (a Stacked_table also counts as one table)
- `as_workbook()` / `save_xlsx()`: To export any of the objects described above to excel workbooks.

### Author(s)

**Maintainer**: Stefan Fleck `<stefan.b.fleck@gmail.com>`

### See Also

Useful links:

- `https://github.com/statistikat/tatoo`
- Report bugs at `https://github.com/statistikat/tatoo/issues`

## `tatoo_table` *Tatoo Table*

### Description

`Tatto_table` is the superclass of all the `*_table` classes made available by this package. Each `Tatoo_table` provides a different way of combining several tables (data.frames) into a single table. Those tables can then be exported via `as_workbook()`/`save_xlsx()`. In the future, support for latex and html export is also planned.

### Usage

```
tatoo_table(dat)
```

### Arguments

| | |
|---|---|
| `dat` | an object of any of the classes listed in the description |

### Details

Currently, the following subclasses exists:

- `Tagged_table`
- `Composite_table`
- `Mashed_table`
- `Stacked_table`

The `tatoo_table()` function is just a constructor used internally and you will not need to use it except if your planning on extending this package with your own code.

### See Also

Other Tatoo tables: `comp_table()`, `mash_table()`, `stack_table()`, `tag_table()`

## `tt_meta` *Tagged Table Metadata*

### Description

Create a `TT_meta` (tagged table metadata) object. In the future, different styling will be supported for title, longtitle and subtitle to make the distinction more meaningful.

### Usage

```
tt_meta(
  table_id = NULL,
  title = NULL,
  longtitle = title,
  subtitle = NULL,
  footer = NULL,
  .print_table_id = FALSE
)
```

### Arguments

| | |
|---|---|
| `table_id` | A scalar (will be coerced to `character`) |
| `title` | A scalar (will be coerced to `character`) |
| `longtitle` | A vector. If `length` > 1 the title will be displayed in several rows |
| `subtitle` | A vector. If `length` > 1 the title will be displayed in several rows |
| `footer` | A vector. If `length` > 1 the title will be displayed in several rows |
| `.print_table_id` | `logical` vector. Whether or not `table_id` should be added to the title of the table in the various output formats. It is recommended to use table_ids only internally (i.e. for `walk_regions()`). |

### Value

a TT_meta object.

### See Also

Tagged_table

---

`vec_prioritise` *Rearrange vector based on priorities*

---

### Description

Shoves elements of a character vector to the front or back. Throws a warning if any elements of ‘high‘ or ‘low‘ are not present in ‘x‘.

### Usage

```
vec_prioritise(x, high = NULL, low = NULL)
```

### Arguments

| | |
|---|---|
| `x` | a character vector |
| `high` | elements to be put to the front |
| `low` | elements to be put to the back |

## Value

a reordered vector

---

`walk_regions` *Apply a function to all named regions on an openxlsx Workbook*

---

## Description

This applies a `.fun` to all named regions in a workbook names match `.pattern`. This is especially useful since `as_workbook()` methods for `Tatoo_tables` add named regions for certain parts of the Table. See also `vignette("named_regions")` for how the names of named regions are constructed by tatoo.

## Usage

```
walk_regions(.wb, .pattern = ".*", .fun, ...)

map_regions(.wb, .pattern = ".*", .fun, ...)
```

## Arguments

| | |
|---|---|
| `.wb` | an openxlsx `Workbook` Object |
| `.pattern` | `character` scalar. A regex filter pattern for named region names (passed on to `grep()`) |
| `.fun` | A function with the formal arguments `wb`, `sheet` and either `rows`, `cols`, or both. For example: `openxlsx::addStyle()`, `openxlsx::addFilter()`, `openxlsx::setRowHeights()`, `openxlsx::setColWidths()` |
| `...` | passed on to `.fun` |

## Value

`walk_regions` returns `.wb`. `map_regions` returns a modified copy of `.wb`

## Examples

```
x <- iris
title(iris) <- "Iris example table"
wb <- as_workbook(iris)

regions(wb)  # display regions


# Apply a style
# Keep in mind that openxlsx functions modify worksheets by reference.
# If you do not want this behaviour you can use map_regions instead.
```

```
style <- openxlsx::createStyle(textDecoration = "bold")
walk_regions(
  wb,
  .pattern = "colnames.*",
  .fun = openxlsx::addStyle,
  style = style
)

## Not run:
  openxlsx::openXL(wb)

## End(Not run)
```

---

`write_worksheet` *Write Data to an openxlsx Worksheet*

---

## Description

This function is similar to `openxlsx::writeData()` from the package, but rather than just writing `data.frames`, `write_worksheet()` supports specialized methods for the various `Tatoo_table` subclasses.

## Usage

```
write_worksheet(
  x,
  wb,
  sheet,
  append = FALSE,
  start_row = 1L,
  ...,
  named_regions = TRUE,
  named_regions_prefix = NA_character_
)

## S3 method for class 'Tagged_table'
write_worksheet(
  x,
  wb,
  sheet = sanitize_excel_sheet_names(attr(x, "meta")$table_id),
  append = FALSE,
  start_row = 1L,
  ...,
  print_table_id = attr(x, "meta")[[".print_table_id"]],
  named_regions = TRUE,
  named_regions_prefix = NA_character_
)

## S3 method for class 'Composite_table'
write_worksheet(
  x,
  wb,
  sheet,
  append = FALSE,
  start_row = 1L,
  ...,
  named_regions = TRUE,
  named_regions_prefix = NA_character_
)

## S3 method for class 'Mashed_table'
write_worksheet(
  x,
  wb,
  sheet,
  append = FALSE,
  start_row = 1L,
  mash_method = attr(x, "mash_method"),
  id_vars = attr(x, "id_vars"),
  insert_blank_row = attr(x, "insert_blank_row"),
  sep_height = attr(x, "sep_height"),
  ...,
  named_regions = TRUE,
  named_regions_prefix = NA_character_
)

## S3 method for class 'Stacked_table'
write_worksheet(
  x,
  wb,
  sheet,
  append = FALSE,
  start_row = 1L,
  spacing = attr(x, "spacing"),
  ...,
  named_regions = TRUE,
  named_regions_prefix = NA_character_
)
```

## Arguments

| | |
|---|---|
| x | A `Tatoo_table`. |
| wb | An openxlsx `Workbook` object |

| | |
|---|---|
| `sheet` | The worksheet to write to. Can be the worksheet index or name. |
| `append` | `logical` Whether or not to append to an existing worksheet or create a new one |
| `start_row` | A scalar integer specifying the starting row to write to. |
| `...` | Additional arguments passed on to methods for overriding the styling attributes of the `Tatoo_tables` you want to export. |
| `named_regions` | `logical`. If `TRUE` (default) named regions are created in the target excel file to identify different parts of the tables (header, body, column names, etc...). These named regions can, for example, be used for applying formats. Creating named regions can be switched of as this might impact performance of the excel conversion and writing of excel files for workbooks with large numbers of tables. |
| `named_regions_prefix` | `character` scalar. Prefix to write in front of all named regions created by `write_worksheet` |
| `print_table_id` | `logical` vector. Whether or not `table_id` should be added to the title of the table. It is recommended to use table_ids only internally (i.e. for `walk_regions()`). |
| `mash_method` | either `"row"` or `"col"`. Should the tables be mashed together with alternating rows or with alternating columns? |
| `id_vars` | If `id_vars` is specified, the tables will be combined using `merge()` on the columns specified in `id_vars`, otherwise the tables will be combined with `cbind()`. |
| `insert_blank_row` | Only if mashing rows: logical. Whether to insert blank rows between mash-groups. *Warning: this converts all columns to character.* Use with care. |
| `sep_height` | Only has an effect when exporting to `xlsx`. if `insert_blank_row == TRUE`, height of the inserted row, else height of the top row of each mash-group. |
| `spacing` | Number of lineskips between the tables when exporting to xlsx |

## Value

an openxlsx Workbook object

## See Also

Other xlsx exporters: `as_workbook()`

# Index