Package 'xltabr'

November 28, 2017

Type Package

Title Automatically Write Beautifully Formatted Cross Tabulations/Contingency Tables to Excel

Version 0.1.2

Description Writes beautifully formatted cross tabulations to Excel using 'openxlsx'. It has been developed to help automate the process of publishing Official Statistics. The user provides a dataframe, which is outputted to Excel with various types of rich formatting which are automatically detected from the structure of the cross tabulation. Documentation can be found at the following url <https://github.com/moj-analytical-services/xltabr>.

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 6.0.1

URL https://github.com/moj-analytical-services/xltabr#readme

BugReports https://github.com/moj-analytical-services/xltabr/issues

Imports magrittr, openxlsx

Suggests testthat, reshape2

NeedsCompilation no

Author Robin Linacre [aut, cre], Karik Isichei [aut]

Maintainer Robin Linacre <robinlinacre@hotmail.com>

Repository CRAN

Date/Publication 2017-11-28 11:26:20 UTC

R topics documented:

add_body *Add a table body (a df) to the tab.*

Description

Add a table body (a df) to the tab.

Usage

```
add_body(tab, df, left_header_colnames = NULL, row_style_names = NULL,
  left_header_style_names = NULL, col_style_names = NULL,
  fill_non_values_with = list(na = NULL, nan = NULL, inf = NULL, neg_inf =
  NULL))
```

Arguments

tab	The core tab object
df	A data frame containing the data you want to write to write to Excel
left_header_colnames	The names of the columns in the df which are left headers, as opposed to the main body
row_style_names	Manually specify the styles that apply to each row (as opposed to using the autodetect functions). Styles provided must be present in the style catalogue
left_header_style_names	Manually specify the addition styles that will apply to cells in the left headers. Must be present in styles catalogue
col_style_names	Manually specify the additional styles that will be applied to each column. Must be present in styles catalogue
fill_non_values_with	Manually specify a list of strings that will replace non numbers types NA, NaN, Inf and -Inf. e.g. list(na = '*', nan = '', inf = '-', neg_inf = '–'). Note: NaNs are not treated as NAs.

Examples

```
crosstab <- read.csv(system.file("extdata", "example_crosstab.csv", package="xltabr"))
tab <- initialise()

# Note you could also use xltabr::auto_detect_left_headers
colnames <- c("drive", "age")
lh_styles <- "left_header"
tab <- add_body(tab, crosstab, left_header_colnames = colnames, left_header_style_names = lh_styles)
```

add_footer — *Add footers to the tab. Footer text is provided as a character vector, with each element being a row of the footer*

Description

Add footers to the tab. Footer text is provided as a character vector, with each element being a row of the footer

Usage

```
add_footer(tab, footer_text, footer_style_names = "footer")
```

Arguments

tab	The core tab object
footer_text	A character vector. Each element is a row of the footer
footer_style_names	A character vector. Each elemment is a style_name

Examples

```
tab <- initialise()
footer_text <- c("Footer contents 1", "Footer contents 2")
footer_style_names <- c("subtitle", "subtitle")
tab <- add_footer(tab, footer_text, footer_style_names)
tab <- write_data_and_styles_to_wb(tab)
```

add_title — *Add titles to the tab. Title text is provided as a character vector, with each element being a row of the title*

Description

Add titles to the tab. Title text is provided as a character vector, with each element being a row of the title

Usage

```
add_title(tab, title_text, title_style_names = NULL)
```

Arguments

tab	The core tab object
title_text	A character vector. Each element is a row of the title
title_style_names	A character vector. Each elemment is a style_name

Examples

```
crosstab <- read.csv(system.file("extdata", "example_crosstab.csv", package="xltabr"))
tab <- initialise()
title_text <- c("Main title on first row", "subtitle on second row")
title_style_names <- c("title", "subtitle")
tab <- add_title(tab, title_text, title_style_names)
```

---

add_top_headers *Add top headers to the tab. The top headers are provided as a character vector. If you need more than one row, provide a list of character vectors. Top headers are automatically assigned the style_text 'top_header_1', but you may provide style overrides using column_style_names and row_style_names*

Description

Add top headers to the tab. The top headers are provided as a character vector. If you need more than one row, provide a list of character vectors. Top headers are automatically assigned the style_text 'top_header_1', but you may provide style overrides using column_style_names and row_style_names

Usage

```
add_top_headers(tab, top_headers, col_style_names = "",
  row_style_names = "body|top_header_1")
```

Arguments

tab	The core tab object
top_headers	For a single top_header row, a character vector. For multiple top_header rows, a list of character vectors.
col_style_names	A character vector, with and element for each column of the top header. Each element is a style name. Col styles in inherit from row_styles.
row_style_names	A character vector, with an element for each row of the top header. Each element is a style_name (i.e. a key in the style catalogue)

Examples

```
crosstab <- read.csv(system.file("extdata", "example_crosstab.csv", package="xltabr"))
tab <- initialise()

top_headers_row_1 <- c("", "", "Car type", "Car type", "Car type")
top_headers_row_2 <- c("Drive", "Age", "Sedan", "Sport", "Supermini")
top_headers <- list(top_headers_row_1, top_headers_row_2)

tab <- add_top_headers(tab, top_headers)
```

`auto_crosstab_to_tab` *Take a cross tabulation produced by* `reshape2::dcast` *and output a formatted openxlsx wb object*

Description

Take a cross tabulation produced by `reshape2::dcast` and output a formatted openxlsx wb object

Usage

```
auto_crosstab_to_tab(df, auto_number_format = TRUE, top_headers = NULL,
  titles = NULL, footers = NULL, indent = TRUE,
  left_header_colnames = NULL, vertical_border = TRUE, auto_merge = TRUE,
  insert_below_tab = NULL, total_text = NULL, include_header_rows = TRUE,
  wb = NULL, ws_name = NULL, number_format_overrides = list(),
  fill_non_values_with = list(na = NULL, nan = NULL, inf = NULL, neg_inf =
  NULL), allcount_to_level_translate = NULL)
```

Arguments

`df`	A data.frame. The cross tabulation to convert to Excel
`auto_number_format`	Whether to automatically detect number format
`top_headers`	A list. Custom top headers. See `add_top_headers()`
`titles`	The title. A character vector. One element per row of title
`footers`	Table footers. A character vector. One element per row of footer.
`indent`	Automatically detect level of indentation of each row
`left_header_colnames`	The names of the columns that you want to designate as left headers
`vertical_border`	Boolean. Do you want a left border?
`auto_merge`	Boolean. Whether to merge cells in the title and footers to width of body
`insert_below_tab`	A existing tab object. If provided, this table will be written on the same sheet, below the provided tab.
`total_text`	The text that is used for the 'grand total' of a cross tabulation
`include_header_rows`	Boolean - whether to include or omit the header rows
`wb`	A existing openxlsx workbook. If not provided, a new one will be created
`ws_name`	The name of the worksheet you want to write to
`number_format_overrides`	e.g. list("colname1" = "currency1") see auto_style_number_formatting

fill_non_values_with
: Manually specify a list of strings that will replace non numbers types NA, NaN, Inf and -Inf. e.g. list(na = '*', nan = '', inf = '-', neg_inf = '–'). Note: NaNs are not treated as NAs.

allcount_to_level_translate
: Manually specify how to translate summary levels into header formatting

Examples

```
crosstab <- read.csv(system.file("extdata", "example_crosstab.csv", package="xltabr"))
tab <- auto_crosstab_to_tab(crosstab)
```

---

auto_crosstab_to_wb *Take a cross tabulation produced by* reshape2::dcast *and output a formatted openxlsx wb object*

Description

Take a cross tabulation produced by reshape2::dcast and output a formatted openxlsx wb object

Usage

```
auto_crosstab_to_wb(df, auto_number_format = TRUE, top_headers = NULL,
  titles = NULL, footers = NULL, auto_open = FALSE, indent = TRUE,
  left_header_colnames = NULL, vertical_border = TRUE, return_tab = FALSE,
  auto_merge = TRUE, insert_below_tab = NULL, total_text = NULL,
  include_header_rows = TRUE, wb = NULL, ws_name = NULL,
  number_format_overrides = list(), fill_non_values_with = list(na = NULL,
  nan = NULL, inf = NULL, neg_inf = NULL), allcount_to_level_translate = NULL,
  left_header_col_widths = NULL, body_header_col_widths = NULL)
```

Arguments

df
: A data.frame. The cross tabulation to convert to Excel

auto_number_format
: Whether to automatically detect number format

top_headers
: A list. Custom top headers. See add_top_headers()

titles
: The title. A character vector. One element per row of title

footers
: Table footers. A character vector. One element per row of footer.

auto_open
: Boolean. Automatically open Excel output.

indent
: Automatically detect level of indentation of each row

left_header_colnames
: The names of the columns that you want to designate as left headers

vertical_border
: Boolean. Do you want a left border?

`return_tab` Boolean. Return a tab object rather than a openxlsx workbook object

`auto_merge` Boolean. Whether to merge cells in the title and footers to width of body

`insert_below_tab`
A existing tab object. If provided, this table will be written on the same sheet, below the provided tab.

`total_text` The text that is used for the 'grand total' of a cross tabulation

`include_header_rows`
Boolean - whether to include or omit the header rows

`wb` A existing openxlsx workbook. If not provided, a new one will be created

`ws_name` The name of the worksheet you want to write to

`number_format_overrides`
e.g. list("colname1" = "currency1") see auto_style_number_formatting

`fill_non_values_with`
Manually specify a list of strings that will replace non numbers types NA, NaN, Inf and -Inf. e.g. list(na = '*', nan = '', inf = '-', neg_inf = '–'). Note: NaNs are not treated as NAs.

`allcount_to_level_translate`
Manually specify how to translate summary levels into header formatting

`left_header_col_widths`
Width of row header columns you wish to set in Excel column width units. If singular, value is applied to all row header columns. If a vector, vector must have length equal to the number of row headers in workbook. Use special case "auto" for automatic sizing. Default (NULL) leaves column widths unchanged.

`body_header_col_widths`
Width of body header columns you wish to set in Excel column width units. If singular, value is applied to all body columns. If a vector, vector must have length equal to the number of body headers in workbook. Use special case "auto" for automatic sizing. Default (NULL) leaves column widths unchanged.

Examples

```
crosstab <- read.csv(system.file("extdata", "example_crosstab.csv", package="xltabr"))
wb <- auto_crosstab_to_wb(crosstab)
```

`auto_detect_body_title_level`
Autodetect the 'title level' of each row in the cross tabulation e.g. title 1 is most prominent, title 2 next etc.

Description

Uses the presence of '(all)' to detect the prominence. The parameter allcount_to_level_translate allows the user to control how the count of '(all)' in the left header is translated into the header level

Usage

```
auto_detect_body_title_level(tab, keyword = "(all)",
  allcount_to_level_translate = NULL)
```

Arguments

tab	a tab object
keyword	The keyword to use to detect summarisation. Uses '(all)' by default because this is what reshape2::dcast uses
allcount_to_level_translate	A named vector that provides a lookup - by default c("0" = NA, "1" = 5, "2" = 4, "3" = 3, "4" = 2, "5" = 1), which says that e.g. allcount 1 results in title_5 etc

Examples

```
crosstab <- read.csv(system.file("extdata", "example_crosstab.csv", package="xltabr"))
tab <- initialise()
tab <- add_body(tab, crosstab, left_header_colnames = c("drive", "age"))
tab <- auto_detect_body_title_level(tab)
```

---

auto_detect_left_headers

Uses the presence of '(all)' in the leftmost columns of data to detect that these columns are really left headers rather than body columns

---

Description

Populates tab$body$left_header_colnames automatically

Usage

```
auto_detect_left_headers(tab, keyword = "(all)")
```

Arguments

tab	a tab object
keyword	The keyword to use to detect summarisation. Uses '(all)' by default because this is what reshape2::dcast uses

Examples

```
crosstab <- read.csv(system.file("extdata", "example_crosstab.csv", package="xltabr"))
tab <- initialise()
tab <- add_body(tab, crosstab)
tab <- auto_detect_left_headers(tab)
```

auto_df_to_wb *Take a data.frame in r and output an openxlsx wb object*

Description

Take a data.frame in r and output an openxlsx wb object

Usage

```
auto_df_to_wb(df, auto_number_format = TRUE, titles = NULL,
  footers = NULL, left_header_colnames = NULL, vertical_border = TRUE,
  auto_open = FALSE, return_tab = FALSE, auto_merge = TRUE,
  insert_below_tab = NULL)
```

Arguments

df	The data.frame to convert to Excel
auto_number_format	Boolean. Whether to automatically detect number formats of columns
titles	Character vector of titles. One element per row of title.
footers	Table footers. A character vector. One element per row of footer.
left_header_colnames	The names of the columns that you want to designate as left headers
vertical_border	Boolean. Do you want a left border?
auto_open	Boolean. Automatically open Excel output.
return_tab	Boolean. Return a tab object rather than a openxlsx workbook object
auto_merge	Boolean. Whether to merge cells in the title and footers to width of body
insert_below_tab	A existing tab object. If provided, this table will be written on the same sheet, below the provided tab.

Examples

```
wb <- auto_crosstab_to_wb(mtcars)
```

auto_merge_footer_cells

Take a tab and merge the footer rows

Description

Take a tab and merge the footer rows

Usage

```
auto_merge_footer_cells(tab)
```

Arguments

tab	The core tab object

Examples

```
crosstab <- read.csv(system.file("extdata", "example_crosstab.csv", package="xltabr"))
tab <- initialise()
tab <- add_body(tab, crosstab)
footer_text <- c("Footer contents 1", "Footer contents 2")
footer_style_names <- c("subtitle", "subtitle")
tab <- add_footer(tab, footer_text, footer_style_names)
tab <- auto_merge_footer_cells(tab)
```

auto_merge_title_cells

Take a tab and merge the title rows

Description

Take a tab and merge the title rows

Usage

```
auto_merge_title_cells(tab)
```

Arguments

tab	The core tab object

Examples

```
crosstab <- read.csv(system.file("extdata", "example_crosstab.csv", package="xltabr"))
tab <- initialise()
tab <- add_body(tab, crosstab)
title_text <- c("Main title on first row", "subtitle on second row")
title_style_names <- c("title", "subtitle")
tab <- add_title(tab, title_text, title_style_names)
tab <- auto_merge_title_cells(tab)
```

`auto_style_indent` *Consolidate the header columns into one, taking the rightmost value and applying indent*

Description

e.g. a | b | (all) -> b e.g. (all) | (all) | (all) -> Grand Total

Usage

```
auto_style_indent(tab, keyword = "(all)", total_text = NULL,
  left_header_colname = " ")
```

Arguments

`tab`	a tab object
`keyword`	The keyword to use to detect summarisation. Uses '(all)' by default because this is what reshape2::dcast uses
`total_text`	The text to use for the grand total (a row where all the left headers are '(all)'. Defaults to Grand Total.
`left_header_colname`	The column name of left header column, which is now a single column.

Examples

```
crosstab <- read.csv(system.file("extdata", "example_crosstab.csv", package="xltabr"))
tab <- initialise()
tab <- add_body(tab, crosstab, left_header_colnames = c("drive", "age"))
tab <- auto_style_indent(tab)
```

auto_style_number_formatting

Use the data type of the columns to choose an automatic Excel format for body cells

Description

This function reads styling from the styles defined here

Usage

```
auto_style_number_formatting(tab, overrides = list())
```

Arguments

tab	a table object
overrides	a list containing any manual overrides where the user wants to provide their own style name

Examples

```
body_data <- readRDS(system.file("extdata", "test_number_types.rds", package="xltabr"))
tab <- initialise()
tab <- add_body(tab, body_data)
tab <- auto_style_number_formatting(tab)
```

body_get_bottom_wb_row

Compute the bottom (lowest) row of the workbook occupied by the body If the body does not exist, returns the last row of the previous element (the top header)

Description

Compute the bottom (lowest) row of the workbook occupied by the body If the body does not exist, returns the last row of the previous element (the top header)

Usage

```
body_get_bottom_wb_row(tab)
```

Arguments

tab	The core tab object

`body_get_cell_styles_table` *Create table with columns |row|col|style name| containing the styles names for each cell of the body*

Description

Create table with columns |row|col|style name| containing the styles names for each cell of the body

Usage

```
body_get_cell_styles_table(tab)
```

Arguments

`tab`	The core tab object

`body_get_rightmost_wb_col` *Compute the rightmost column of the workbook which is occupied by the body*

Description

Compute the rightmost column of the workbook which is occupied by the body

Usage

```
body_get_rightmost_wb_col(tab)
```

Arguments

`tab`	The core tab object

body_get_wb_cols *Compute the columns of the workbook which are occupied by the body*

Description

Compute the columns of the workbook which are occupied by the body

Usage

```
body_get_wb_cols(tab)
```

Arguments

tab	The core tab object

body_get_wb_left_header_cols *Compute the columns of the workbook which are occupied by the left header columns (which are a subset of the body columns)*

Description

Compute the columns of the workbook which are occupied by the left header columns (which are a subset of the body columns)

Usage

```
body_get_wb_left_header_cols(tab)
```

Arguments

tab	The core tab object

body_get_wb_rows *Compute the rows of the workbook which are occupied by the body*

Description

Compute the rows of the workbook which are occupied by the body

Usage

```
body_get_wb_rows(tab)
```

Arguments

tab	The core tab object

`body_initialise` *Initialise the body element of the tab, creating all the required properties the core tab object*

Description

Initialise the body element of the tab, creating all the required properties the core tab object

Usage

```
body_initialise(tab)
```

Arguments

`tab`	The core tab object

`body_write_rows` *Write all the body data to the workbook (but do not write style information)*

Description

Write all the body data to the workbook (but do not write style information)

Usage

```
body_write_rows(tab)
```

Arguments

`tab`	The core tab object

`get_cell_format_path` *Get cell formats path*

Description

Returns path to cell format definitions which is set in global options. To change this path see set_cell_format_path. Default cell format used by package is here

Usage

```
get_cell_format_path()
```

Examples

```
get_cell_format_path()
```

get_num_format_path *Get number formats path*

Description

Returns path to number format definitions which is set in global options. To change this path see get_num_format_path. Default number formats used by package is here

Usage

```
get_num_format_path()
```

Examples

```
get_num_format_path()
```

get_style_path *Get style sheet path*

Description

Returns path to current style sheet which is set in global options. To change this path see set_style_path. Default stylesheet used by package is here

Usage

```
get_style_path()
```

Examples

```
get_style_path()
```

initialise *Create a new xltabr object for cross tabulation*

Description

Create a new xltabr object for cross tabulation

Usage

```
initialise(wb = NULL, ws_name = NULL, topleft_row = 1, topleft_col = 1,
  insert_below_tab = NULL)
```

Arguments

wb	An openxlsx workbook object to write to. If null, a new one will be created
ws_name	The worksheet to write to. If null, Sheet1 will be used
topleft_row	Specifies the row where the table begins in the worksheet
topleft_col	Specifies the column where the table begins in the worksheet
insert_below_tab	If given, the new tab will be inserted immediately below this one

Value

A list which contains the dataframe

Examples

```
tab <- initialise()
```

---

set_cell_format_path *Set cell format path*

Description

Set the path to the cell formats to be used by xltabr. To get this path see get_cell_format_path. Default cell format used by package is here. If no path is supplied the function sets the cell format to default to default.

Usage

```
set_cell_format_path(path = NULL)
```

Arguments

path	the file path to the cell formats (csv file). If NULL the function sets the cell format to the default option.

Examples

```
my_file_path <- system.file("extdata", "style_to_excel_number_format.csv", package = "xltabr")
set_cell_format_path(my_file_path)
```

set_num_format_path *Set number format path*

Description

Set the path to the number formats to be used by xltabr. To get this path see get_num_format_path. Default number format used by package is here. If no path is supplied the function sets the cell format to default to default.

Usage

```
set_num_format_path(path = NULL)
```

Arguments

path	the file path to the number formats (csv file). If NULL the function sets the number format to the default option.

Examples

```
my_file_path <- system.file("extdata", "number_format_defaults.csv", package = "xltabr")
set_num_format_path(my_file_path)
```

set_style_path *Set style sheet path*

Description

Set the path to the style sheet to be used by package. To get this path see get_style_path. Default cell formats used by xltabr is here. If no path is supplied the function sets the style sheet to default.

Usage

```
set_style_path(path = NULL)
```

Arguments

path	the file path to the style sheet path (xlsx file). If NULL the function sets the cell format to the default option.

Examples

```
my_file_path <- system.file("extdata", "styles.xlsx", package = "xltabr")
set_style_path(my_file_path)
```

set_wb_widths *Set column widths to tab workbook*

Description

Set column widths to tab workbook

Usage

```
set_wb_widths(tab, left_header_col_widths = NULL,
  body_header_col_widths = NULL)
```

Arguments

tab	The core tab object
left_header_col_widths	Width of row header columns you wish to set in Excel column width units. If singular, value is applied to all row header columns. If a vector, vector must have length equal to the number of row headers in workbook. Use special case "auto" for automatic sizing. Default (NULL) leaves column widths unchanged.
body_header_col_widths	Width of body header columns you wish to set in Excel column width units. If singular, value is applied to all body columns. If a vector, vector must have length equal to the number of body headers in workbook. Use special case "auto" for automatic sizing. Default (NULL) leaves column widths unchanged.

Examples

```
crosstab <- read.csv(system.file("extdata", "example_crosstab.csv", package="xltabr"))
tab <- initialise()
colnames <- c("drive", "age")
tab <- add_body(tab, crosstab, left_header_colnames = colnames)
tab <- set_wb_widths(tab, left_header_col_widths = "auto", body_header_col_widths = c(7,14,28))
```

style_catalogue_add_excel_num_format *Manually add an Excel num format to the style catalogue*

Description

Manually add an Excel num format to the style catalogue

Usage

```
style_catalogue_add_excel_num_format(tab, style_string, excel_num_format)
```

Arguments

tab	a table object
style_string	the name (key) in the tab$style_catalogue
excel_num_format	an excel number format e.g. "#.00"

Examples

```
tab <- xltabr::initialise()
tab <- style_catalogue_add_excel_num_format(tab, "currency2", "£ #,###")
```

---

style_catalogue_add_openxlsx_style

Manually add an openxlsx s4 style the style catalogue

Description

Manually add an openxlsx s4 style the style catalogue

Usage

```
style_catalogue_add_openxlsx_style(tab, style_string, openxlsx_style,
  row_height = NULL)
```

Arguments

tab	a table object
style_string	the name (key) in the tab$style_catalogue
openxlsx_style	an openxlsx s4 style
row_height	the height of the row. optional.

Examples

```
tab <- xltabr::initialise()
s4style <- openxlsx::createStyle(fontName = "Courier",
                                 fontColour = "#80a9ed",
                                 fontSize = 20,
                                 numFmt =  "£    #,###")
tab <- style_catalogue_add_openxlsx_style(tab, "custom", s4style, row_height = 40)
```

`title_get_bottom_wb_row`

Get the bottom row which the titles occupy in the workbook If no titles have been provided, return the cell above the topleft row of the extent. The next element (top headers, body or whatever it is) will want to be positioned in the cell below this

---

Description

Get the bottom row which the titles occupy in the workbook If no titles have been provided, return the cell above the topleft row of the extent. The next element (top headers, body or whatever it is) will want to be positioned in the cell below this

Usage

```
title_get_bottom_wb_row(tab)
```

Arguments

`tab`	The core tab object

`title_get_cell_styles_table`

Create table with columns \|row\|col\|style name\| that contains the styles names

---

Description

Create table with columns |row|col|style name| that contains the styles names

Usage

```
title_get_cell_styles_table(tab)
```

Arguments

`tab`	The core tab object

title_get_rightmost_wb_col *Get the rightmost column occupied by the titles in the workbook*

Description

Get the rightmost column occupied by the titles in the workbook

Usage

```
title_get_rightmost_wb_col(tab)
```

Arguments

tab	The core tab object

title_get_wb_cols *Get the columns occupied by the title in the workbook*

Description

Get the columns occupied by the title in the workbook

Usage

```
title_get_wb_cols(tab)
```

Arguments

tab	The core tab object

title_get_wb_rows *Get the rows occupied by the title in the workbook*

Description

Get the rows occupied by the title in the workbook

Usage

```
title_get_wb_rows(tab)
```

Arguments

tab	The core tab object

`title_initialise` *Create all the required properties for the title on the tab object*

Description

Create all the required properties for the title on the tab object

Usage

```
title_initialise(tab)
```

Arguments

`tab`	core tab object

`title_write_rows` *Write all the title data to the workbook (but do not write style data)*

Description

Write all the title data to the workbook (but do not write style data)

Usage

```
title_write_rows(tab)
```

Arguments

`tab`	The core tab object

`top_headers_get_bottom_wb_row` *Compute the bottom (lowest) row of the workbook occupied by the top headers If the top headers do not exist, returns the last row of the previous element (the titles)*

Description

Compute the bottom (lowest) row of the workbook occupied by the top headers If the top headers do not exist, returns the last row of the previous element (the titles)

Usage

```
top_headers_get_bottom_wb_row(tab)
```

Arguments

tab	The core tab object

`top_headers_get_cell_styles_table`

Create table with columns |row|col|style name| containing the styles names for each cell of the top headers

---

Description

Create table with columns |row|col|style name| containing the styles names for each cell of the top headers

Usage

```
top_headers_get_cell_styles_table(tab)
```

Arguments

tab	The core tab object

`top_headers_get_rightmost_wb_col`

Compute the rightmost column of the workbook which is occupied by the top headers

---

Description

Compute the rightmost column of the workbook which is occupied by the top headers

Usage

```
top_headers_get_rightmost_wb_col(tab)
```

Arguments

tab	The core tab object

`top_headers_get_wb_cols` *Compute the columns of the workbook which are occupied by the top headers*

Description

Compute the columns of the workbook which are occupied by the top headers

Usage

```
top_headers_get_wb_cols(tab)
```

Arguments

`tab`	The core tab object

`top_headers_get_wb_rows` *Compute the rows of the workbook which are occupied by the top headers*

Description

Compute the rows of the workbook which are occupied by the top headers

Usage

```
top_headers_get_wb_rows(tab)
```

Arguments

`tab`	The core tab object

top_headers_initialise

Create all the required properties for the top headers on the tab object

Description

Create all the required properties for the top headers on the tab object

Usage

```
top_headers_initialise(tab)
```

Arguments

tab	The core tab object

top_headers_write_rows

Write all the top header data to the workbook (but do not write style information)

Description

Write all the top header data to the workbook (but do not write style information)

Usage

```
top_headers_write_rows(tab)
```

Arguments

tab	The core tab object

write_data_and_styles_to_wb *Write the styles and data contained in tab object to the workbook at tab$wb. This is useful if the user has prepared a tab object, e.g. using auto_crosstab_to_tab maybe performed some customisation, and now needs to write out to the workbook*

Description

Write the styles and data contained in tab object to the workbook at tab$wb. This is useful if the user has prepared a tab object, e.g. using auto_crosstab_to_tab maybe performed some customisation, and now needs to write out to the workbook

Usage

```
write_data_and_styles_to_wb(tab)
```

Arguments

tab	a table object

Examples

```
crosstab <- read.csv(system.file("extdata", "example_crosstab.csv", package="xltabr"))
tab <- auto_crosstab_to_tab(crosstab)
tab <- write_data_and_styles_to_wb(tab)
```

Index