Package: xldiff (via r-universe)

November 1, 2024

Title Compare excel sheets

Version 0.0.0.9000

Description `xldiff` provides tools to compare excel sheets, broadly inspired by ``diff"-type functions. Provided functions can read sheets of two excel files and produce a third file that highlights cells that have changed. In the case of numeric changes, the direction of change is highlighted. These tools do not account for structural changes in the sheets (e.g., the addition of a column), but are useful in tracking changed values in tables or parameter files. Utility functions developed to streamline formatting output files are also more broadly useful in programmatically formatting excel files using openxlsx.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

Imports cli, dplyr, methods, openxlsx, purrr, readxl, rlang

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

RemoteUrl https://github.com/cbedwards-dfw/xldiff

RemoteRef HEAD

RemoteSha ec0e55068203cc9370f4edc266ee0ba00899a423

Contents

`add_cell_borders` *Adds cell borders to openxlsx spreadsheet*

Description

When calling `sheet_diff()`, creating a new workbook for the diff contents, and then coloring to highlight changed cells, the original spreadsheet formatting is lost. To facilitate interpretting the diff, it can be useful to recreate the major components of the original formatting, especially cell borders. This function adds cell borders, and is designed for ease of use when replicating formatting from the original excel file. Blocks of cells to give borders to can be specified in the original excel format (e.g. "A1:D5"). For only outside borders around each block (default), use argument `every.cell = FALSE`. To add all the cell borders within each block to generate a grid appearance, set `every.cell = TRUE`. Note that non-border formatting of each cell will not be maintained, but border formatting will be overwritten. When adding thin boundaries between inner cells and a thick outer border for a block of cells, first use `add_cell_borders` to with `every.cell = TRUE`, and appropriate border arguments (usually `border.thickness = "thin"`) and then use again with `every.cell = FALSE` and appropriate border arguments (usually `border.thickness = "medium"`).

Usage

```
add_cell_borders(
  wb,
  sheet,
  block.ranges,
  sheet.start = "A1",
  every.cell = FALSE,
  border.col = "black",
  border.thickness = "medium"
)
```

Arguments

`wb`	openxlsx workbook object
`sheet`	character corresponding to sheet name of openxlsx workbook object `wb`.
`block.ranges`	One or more cell ranges specified in excel format (e.g. `c("A1:D5, "B6", "A8:D8")`)
`sheet.start`	Optional. If `wb$sheet` corresponds to an excel sheet in which the `wb$sheet` entries were read starting on a cell other than "A1" (e.g. `readxl::read_excel` with range specified or skip provided), provide the top left cell that was read into R in order to handle the offsetting, so that you can specify cell ranges based on the original excel file.
`every.cell`	Do we want borders around each individual cell in each cell block (`TRUE`), or just around the outer edges of the block (`FALSE`). Defaults to `FALSE`.
`border.col`	Color for border. See `?openxlsx::createStyle` for details. Defaults to "black".

border.thickness
: Thickness for border. See ?openxlsx::createStyle for details. Common choices: "thin", "thick".

---

add_changed_formats *Format* openxlsx *worksheet based on changes*

Description

Highlights cells that changed, coloring differently for increasing, decreasing, and non-numeric cells. Typically used on a worksheet that contains the $sheet.diff dataframe from the same sheet comparison as provided in the cur.sheet argument. In some cases it may be useful to define custom color schemes (e.g., if increasing numbers are good and decreasing numbers are bad, you may want green and red foregrounds for those types of changes). Individual styles can be provided with optional arguments see ?openxlsx::createStyle for options in defining styles.

Usage

```
add_changed_formats(
  wb,
  cur.sheet,
  sheet.comp,
  rows.invert = NULL,
  cols.invert = NULL,
  df.invert = NULL,
  nofillStyle = NULL,
  changeStyle = NULL,
  posStyle = NULL,
  negStyle = NULL
)
```

Arguments

wb	openxlsx workbook to make on which to make changes.
cur.sheet	sheet name in openxlsx workbook on which to make changes.
sheet.comp	list of comparison dataframes generated by sheet_comp()
rows.invert	Optional vector of row numbers to invert color scheme for increase vs decrease.
cols.invert	Optional vector of column numbers to invert color scheme for increase vs decrease.
df.invert	Optional data frame with $row and $col entries to indentify individual cells to invert color schemes for increase vs decrease. Defaults to NULL.
nofillStyle	Optional openxlsx style object for cells with no changes flagged. Default has black text, white foreground. Create custom style with openxlsx::createStyle().
changeStyle	As nofillStyle, but for non-numeric cells with changed values. Default has black text, light purple forecround.

posStyle	As nofillStyle, but for numeric cells that increase in value. Default has black text, light coral foreground.
negStyle	As nofillStyle, but for numeric cells that decrease in value. Default has black text, light green foreground.

Details

In some cases it may make sense to reverse the color scheme of numeric changes for individual rows, columns, or cells (e.g., when scanning fishery model outputs, increasing fish escapement and decreasing fish exploitation rates logically should both show the same color. Similarly, increasing profits and decreasing costs logically should both show the same color.). Optional arguments rows.invert, cols.invert, and df.invert allow you to specify individual regions of the dataframe to reverse the color scheme.

---

cells_stylize *Apply style to worksheet based on one or more excel-style cell ranges*

Description

Apply style to worksheet based on one or more excel-style cell ranges

Usage

```
cells_stylize(wb, sheet, style, block.ranges, stack = TRUE)
```

Arguments

wb	openxlsx workbook object
sheet	character corresponding to sheet name of openxlsx workbook object wb.
style	openxlsx cell style, created with openxlsx::createStyle(). This can include text size, bolding or italics, text wrapping, foreground color, text color, etc. See ?openxlsx::createStyle for details.
block.ranges	One or more cell ranges specified in excel format (e.g. c("A1:D5, "B6", "A8:D8"))
stack	Should style be appended to existing styles (TRUE) or replace existing styles (FALSE). Defaults to TRUE.

cell_range_translate *Translates from excel cell address to rows and columns*

Description

Translates from excel cell address to rows and columns

Usage

```
cell_range_translate(x, expand = TRUE, start = "A1")
```

Arguments

x	Single string of individual cell or cell range (e.g. "D6" or "D6:AC8")
expand	If TRUE (default), provides the row and column for all cells in the range. If False, provides just the row and column of the start and end cells.
start	Optional argument to account for offset when matching cells in an excel file to a dataframe when the dataframe was generated by reading the excel file starting at a location other than "A1".

Value

dataframe of addresses for each cell in the range, where $row gives the row number and $col gives the column number.

Examples

```
cell_range_translate("D6")
cell_range_translate("A2:H3")
```

excel_diff *Minimal spreadsheet comparison function*

Description

Compares a single sheet between two files, supports providing additional formatting in the form of the optional extra_format_fun argument. For more complex use cases (e.g., multiple sheet, pre-comparison formatting to compare only specific regions, etc) excel_diff can be used as a simple template for writing your own function.

Usage

```
excel_diff(
  file.1,
  file.2,
  results.name,
  sheet.name,
  extra_format_fun = NULL,
  ...
)
```

Arguments

file.1	Filename (including path) for first file to compare
file.2	Filename (including path) for second file to compare
results.name	Name (including path) for fille to save comparison to. Must end in ".xlsx"
sheet.name	character string of sheet to compare (must be present in both files)
extra_format_fun	Optional function to apply additional formatting, allowing users to specify additional calls of addStyle() (or other openxslx functions, like setting column width). First argument must be the workbook object this function makes changes to; second argument must be the name of the worksheet this function makes changes to
...	Additional arguments passed to extra_format_fun

Examples

```
## Not run:
filename.1 = "Documents/WDFW FRAM team work/NOF material/NOF 2024/FRAM/Chin1124.xlsx"
filename.2 = "Documents/WDFW FRAM team work/NOF material/NOF 2024/NOF 2/Chin2524.xlsx"

excel_diff(file.1 = filename.1,
          file.2 = filename.2,
         results.name = "Documents/WDFW FRAM team work/NOF material/NOF 2024/test1.xlsx",
          sheet.name = "ER_ESC_Overview_New"
)

## create function to add in some additional formatting:
extra_form_fun = function(wb, sheet){
 ## add bold and increased size for the first two rows.
 openxlsx::addStyle(wb, sheet,
                  style = openxlsx::createStyle(fontSize = 16, textDecoration = "Bold"),
                    rows = 1:2, cols = 1:8, gridExpand = TRUE,
                    stack = TRUE)
 ## add thin inner cell borders
 add_cell_borders(wb, sheet,
                  block.ranges = c("B3:H34") )
 ## add thick outer borders
 add_cell_borders(wb, sheet,
                  block.ranges = c("A2", "B1:D2", "E1:H2",
```

```
                                    "A3:A34", "B3:D34", "E3:H34",
                                    "D36:H37"),
                    border.thickness = "medium")
}

excel_diff(file.1 = filename.1,
           file.2 = filename.2,
          results.name = "Documents/WDFW FRAM team work/NOF material/NOF 2024/test2.xlsx",
           sheet.name = "ER_ESC_Overview_New",
           extra_format_fun = extra_form_fun
)

## End(Not run)
```

`sheet_comp` *Compare two dataframes of spreadsheets*

Description

Primary funtion for `xldiff` package. When cell values change between dataframe `t1` and dataframe `t2`, the corresponding `$sheetdiff` entry will show [the first value] `-->` [the second value]. Note that because these changes are presenting as characters, changes in numbers with many digits can produce difficult-to-read cells. The `digits.signif` can be used to determine how many significant digits should be presented in the "arrow" cells.

Usage

```
sheet_comp(t1, t2, digits.signif = 4)
```

Arguments

`t1`	First dataframe
`t2`	Second dataframe, same dimensions as first.
`digits.signif`	When flagging changes, comparison is presented in character form. How many significant digits do we present for numerical entries? Numeric, defaults to 4.

Value

List of comparison data frames, including logical matrices used in formatting cells to highlight changes.

- `$sheet.diff`: cell entries for comparison
- `$mat.changed` logical matrix where `TRUE` corresponds to a cell that changed
- `$mat.diff.decrease`: logical matrix where `TRUE` corresponds to a cell of numeric values that decreased = `mat.diff.increase`: as above, but for increases.

Examples

```
## Not run:
## using palmerpenguins data to simulate spreadsheets
library(palmerpenguins)
t1 = t2 = head(penguins)
## change island variable to characters for easier modification
t2$island = t1$island = as.character(t1$island)
## change several entries in the second version
t2$island[3] = "Scotland"
t2$flipper_length_mm[1] = 18
sheet_comp(t1, t2, digits.signif = 4)

## End(Not run)
```

Index