Package 'reactablefmtr'

March 18, 2021

Type Package

Title Simplify Formatting of Tables Made with 'Reactable'

Version 0.2.0

Maintainer Kyle Cuilla <kyle.cuilla@gmail.com>

Description Streamlines the process of applying formats and styles to tables using 'reactable'. Style your 'reactable' table with conditional formatting, color scales, data bars, and icons.

URL https://kcuilla.github.io/reactablefmtr/, https://github.com/kcuilla/reactablefmtr

BugReports https://github.com/kcuilla/reactablefmtr/issues

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Depends R (>= 3.5.0), reactable (>= 0.2.0)

Imports dplyr, grDevices, htmltools, shiny, stringr

Suggests MASS, scales

RoxygenNote 7.1.1

Language en-US

NeedsCompilation no

Author Kyle Cuilla [aut, cre, cph], Greg Lin [ctb], June Choe [ctb]

Repository CRAN

Date/Publication 2021-03-18 19:50:02 UTC

R topics documented:

`color_scales` *Add color scales to rows in a column*

Description

The 'color_scales()' function conditionally colors each cell of a column depending on their value in relation to other values in that particular column. It should be placed within the style argument in reactable::colDef.

Usage

```
color_scales(
  data,
  colors = c("#ff3030", "#ffffff", "#1e90ff"),
  bright_values = TRUE,
  span = FALSE
)
```

Arguments

`data`	Dataset containing at least one numeric column.
`colors`	A vector of colors to color the cells. Colors should be given in order from low values to high values. Default colors provided are red-white-blue: c("#ff3030", "#ffffff", "#1e90ff"). Can use R's built-in colors or other color packages.
`bright_values`	Optionally display values as white. Values with a dark-colored background will be shown in white. Default is set to TRUE but can be turned off by setting to FALSE.
`span`	Optionally apply colors to values across multiple columns instead of by each column. To apply across all columns set to TRUE. If applying to a set of columns, can provide either column names or column positions. Default is set to FALSE.

Value

a function that applies conditional colors to a column of numeric values.

Examples

```
data <- iris[10:29, ]

## By default, the colors_scales() function uses a red-white-blue three-color pattern
reactable(data,
 columns = list(
 Petal.Length = colDef(style = color_scales(data))))

## If only two colors are desired,
## you can specify them with colors = 'c(color1, color2)';
reactable(data,
 columns = list(
 Petal.Length = colDef(style = color_scales(data,
 colors = c("red", "green")))))

## Apply color_scales() across all numeric columns using reactable::defaultColDef
reactable(data,
defaultColDef = colDef(style = color_scales(data)))

## Use span to apply colors to values in relation to the entire dataset
reactable(data,
defaultColDef = colDef(style = color_scales(data, span = TRUE)))

## Span can take column names
reactable(data,
defaultColDef = colDef(style = color_scales(data, span = c("Sepal.Length", "Sepal.Width"))))

## Or it can also take column positions instead
reactable(data,
defaultColDef = colDef(style = color_scales(data, span = 1:2)))
```

`color_tiles` *Add color tiles to rows in a column*

Description

The 'color_tiles()' function conditionally colors the background of each cell similarly to color_scales(). The difference is that color_tiles() uses round colored tiles around values instead of the entire background of the cell. Another difference is color_tiles() allows number formatting with number_fmt whereas color_scales() does not. The last difference is it needs to placed within the cell argument in reactable::colDef vs the style argument for color_scales().

Usage

```
color_tiles(
  data,
  colors = c("#ff3030", "#ffffff", "#1e90ff"),
  number_fmt = NULL,
```

```
  bright_values = TRUE,
  span = FALSE
)
```

Arguments

data	Dataset containing at least one numeric column.
colors	A vector of colors to color the cells. Colors should be given in order from low values to high values. Default colors provided are red-white-blue: c("#ff3030", "#ffffff", "#1e90ff"). Can use R's built-in colors or other color packages.
number_fmt	Optionally format numbers using formats from the scales package. Default is set to NULL.
bright_values	Optionally display values as white. Values with a dark-colored background will be shown in white. Default is set to TRUE but can be turned off by setting to FALSE.
span	Optionally apply colors to values across multiple columns instead of by each column. To apply across all columns set to TRUE. If applying to a set of columns, can provide either column names or column positions. Default is set to FALSE.

Value

a function that applies conditional color tiles to a column of numeric values.

Examples

```
data <- iris[10:29, ]

## By default, the colors_tiles() function uses a red-white-blue three-color pattern
reactable(data,
 columns = list(
 Petal.Length = colDef(cell = color_tiles(data))))

## If only two colors are desired,
## you can specify them with colors = 'c(color1, color2)';
reactable(data,
 columns = list(
 Petal.Length = colDef(cell = color_tiles(data,
 colors = c("red", "green")))))

## Use span to apply colors to values in relation to the entire dataset
reactable(data,
defaultColDef = colDef(cell = color_tiles(data, span = TRUE)))

## Use number_fmt to format numbers using the scales package
car_prices <- MASS::Cars93[20:49, c("Make", "Price")]

reactable(car_prices,
defaultColDef = colDef(cell = color_tiles(car_prices,
number_fmt = scales::dollar)))
```

```
## Use span to apply colors to values in relation to the entire dataset
reactable(data,
defaultColDef = colDef(cell = color_tiles(data, span = TRUE)))

## Span can take column names
reactable(data,
defaultColDef = colDef(cell = color_tiles(data, span = c("Sepal.Length", "Sepal.Width"))))

## Or it can also take column positions instead
reactable(data,
defaultColDef = colDef(cell = color_tiles(data, span = 1:2)))
```

`data_bars` *Add horizontal bars to rows in a column*

Description

The ‘data_bars()‘ function conditionally adds a horizontal bar to each row of a column. The length of the bars are relative to the value of the row in relation to other values within the same column. It should be placed within the cell argument in reactable::colDef.

Usage

```
data_bars(data, colors = "#1e90ff", background = "white", number_fmt = NULL)
```

Arguments

`data`	Dataset containing at least one numeric column.
`colors`	A single color or a vector of colors. Colors should be given in order from low values to high values. Can use R's built-in colors or other color packages.
`background`	Optionally assign a color to use as the background for cells. Default is set to white.
`number_fmt`	Optionally format numbers using formats from the scales package. Default is set to NULL.

Value

a function that applies data bars to a column of numeric values.

Examples

```
library(reactable)
data <- MASS::Cars93[20:49, c("Make", "MPG.city", "MPG.highway")]

## Horizontal bars with lengths relative to cell value
reactable(data,
```

```
columns = list(
MPG.city = colDef(
name = "MPG (city)",
align = "left",
cell = data_bars(data, "dodgerblue"))))

## Add background color
reactable(data,
columns = list(
MPG.city = colDef(
name = "MPG (city)",
align = "left",
cell = data_bars(data, "dodgerblue", "grey"))))

## Conditionally color data bars based on their relative values
## by supplying more than one color
## and apply across all numeric columns using reactable::defaultColDef
reactable(data,
pagination = FALSE,
defaultSortOrder = "desc",
defaultSorted = "MPG.city",
defaultColDef = colDef(
cell = data_bars(data,
colors = c("firebrick1","gold","limegreen"))))

## Use number_fmt to format numbers using the scales package
car_prices <- MASS::Cars93[20:49, c("Make", "Price")]

reactable(car_prices,
defaultColDef = colDef(cell = data_bars(car_prices,
number_fmt = scales::dollar_format(accuracy = 0.1))))
```

`data_bars_gradient` *Add horizontal gradient bars to rows in a column*

Description

The ‘data_bars_gradient()‘ function conditionally adds a left-to-right linear gradient to each row of a column. The length of the bars are relative to the value of the row in relation to other values within the same column. It should be placed within the cell argument in reactable::colDef.

Usage

```
data_bars_gradient(
  data,
  colors = c("#1efffd", "#1e20ff"),
  background = "white",
  number_fmt = NULL
)
```

Arguments

data	Dataset containing at least one numeric column.
colors	A vector of colors of at least two colors. Colors should be given in order from left to right as shown on the data bar. Default colors are c("#1efffd", "#1e20ff").
background	Optionally assign a color to use as the background for cells. Default is set to white.
number_fmt	Optionally format numbers using formats from the scales package. Default is set to NULL.

Value

a function that applies data bars to a column of numeric values.

Examples

```
library(reactable)
data <- MASS::Cars93[20:49, c("Make", "MPG.city", "MPG.highway")]

## By default, colors are provided
reactable(data,
defaultColDef = colDef(
align = "left",
cell = data_bars_gradient(data)))

## Apply custom colors for gradient
reactable(data,
defaultColDef = colDef(
align = "left",
cell = data_bars_gradient(data, colors = c("white","grey","black"))))

## Apply bacground color
reactable(data,
defaultColDef = colDef(
align = "left",
cell = data_bars_gradient(data, background = "black")))

## Use number_fmt to format numbers using the scales package
car_prices <- MASS::Cars93[20:49, c("Make", "Price")]

reactable(car_prices,
defaultColDef = colDef(cell = data_bars_gradient(car_prices,
number_fmt = scales::dollar_format(accuracy = 0.1))))
```

---

data_bars_pos_neg *Add horizontal bars to rows in a column containing positive and negative values*

Description

The 'data_bars_pos_neg()' function conditionally adds a negative horizontal bar to each row of a column containing negative values, and a positive horizontal bar to each row containing positive values. The length of the bars are relative to the value of the row in relation to other values within the same column. It should be placed within the cell argument in reactable::colDef.

Usage

```
data_bars_pos_neg(data, colors = c("red", "green"), number_fmt = NULL)
```

Arguments

data	Dataset containing at least one numeric column.
colors	A minimum of two colors or a vector of colors. Colors should be given in order from negative values to positive values. Can use R's built-in colors or other color packages.
number_fmt	Optionally format numbers using formats from the scales package. Default is set to NULL.

Value

a function that applies positive and negative data bars to a column of numeric values.

Examples

```
data <- data.frame(
company = sprintf("Company%02d", 1:10),
profit_chg = c(0.2, 0.685, 0.917, 0.284, 0.105, -0.701, -0.528, -0.808, -0.957, -0.11))

## By default, the negative values are assigned a red bar,
## and the positive values are assigned a green bar
reactable(data,
bordered = TRUE,
columns = list(
 company = colDef(name = "Company",
 minWidth = 100),
 profit_chg = colDef(
   name = "Change in Profit",
   defaultSortOrder = "desc",
   align = "center",
   minWidth = 400,
   cell = data_bars_pos_neg(data))))

## You can apply a relative color scale to the bars by assigning three or more colors
reactable(data,
bordered = TRUE,
columns = list(
  company = colDef(name = "Company",
  minWidth = 100),
  profit_chg = colDef(
```

```
  name = "Change in Profit",
  defaultSortOrder = "desc",
  align = "center",
  minWidth = 400,
  cell = data_bars_pos_neg(data,
  colors = c("#ff3030", "#ffffff", "#1e90ff")))))

## Use number_fmt to format numbers using the scales package
reactable(data,
bordered = TRUE,
columns = list(
  company = colDef(name = "Company",
  minWidth = 100),
  profit_chg = colDef(
  name = "Change in Profit",
  defaultSortOrder = "desc",
  align = "center",
  minWidth = 400,
  cell = data_bars_pos_neg(data,
  colors = c("#ff3030", "#ffffff", "#1e90ff"),
  number_fmt = scales::percent))))
```

`embed_img` *Embed image from web to rows in a column*

Description

The 'embed_img()' function adds images obtained from the web to a column within reactable. It should be placed within the cell argument in reactable::colDef.

Usage

```
embed_img(data, height = "24", width = "24", label = NULL)
```

Arguments

`data`	Dataset containing URL's to images
`height`	A value given for the height of the image in px. Default height is 24px.
`width`	A value given for the width of the image in px. Default width is 24px.
`label`	Optionally assign a label to the image from another column. Default is set to NULL or no label.

Value

a function that renders an image to a column containing a valid web link.

Examples

```
## If no image links are in the original dataset, you need to assign them like so:
library(dplyr)
data <- iris %>%
 mutate(
 img = case_when(
 Species == "setosa" ~
 "https://upload.wikimedia.org/wikipedia/commons/d/d9/Wild_iris_flower_iris_setosa.jpg",
 Species == "versicolor" ~
 "https://upload.wikimedia.org/wikipedia/commons/7/7a/Iris_versicolor.jpg",
 Species == "virginica" ~
 "https://upload.wikimedia.org/wikipedia/commons/9/9f/Iris_virginica.jpg",
 TRUE ~ "NA"))

## Then use embed_img() to display images
reactable(data,
columns = list(
 img = colDef(cell = embed_img())))

## By default, images are given a size of 24px by 24px,
## but you can adjust the size using height and width:
reactable(data,
columns = list(
 img = colDef(cell = embed_img(height = "50", width = "45"))))

## Optionally assign a label to the image from another column
reactable(data,
columns = list(
 img = colDef(cell = embed_img(data, label = "Species"))))
```

`highlight_max` *Highlights the maximum value in a column*

Description

The 'highlight_max()' function assigns a font color and/or background color to the maximum value in a column. It should be placed within the style argument in reactable::colDef.

Usage

```
highlight_max(data, font_color = "green", highlighter = NULL)
```

Arguments

`data`	Dataset containing at least one numeric column.
`font_color`	color to assign to maximum value in a column. Default color is green.
`highlighter`	color to assign the background of a cell containing maximum value in a column.

Value

a function that applies a color to the maximum value in a column of numeric values.

Examples

```
data <- MASS::road[11:17, ]

## By default, the maximum value is bold with a green font color
reactable(data,
defaultColDef = colDef(
    style = highlight_max(data)))

## Assign a different font color
reactable(data,
defaultColDef = colDef(
    style = highlight_max(data,
    font_color = "red")))

## Highlight the background of the cell for the maximum value in each column
reactable(data,
defaultColDef = colDef(
    style = highlight_max(data,
    highlighter = "yellow")))
```

---

highlight_min *Highlights the minimum value in a column*

Description

The 'highlight_min()' function assigns a font color and/or background color to the minimum value in a column. It should be placed within the style argument in reactable::colDef.

Usage

```
highlight_min(data, font_color = "red", highlighter = NULL)
```

Arguments

data	Dataset containing at least one numeric column.
font_color	color to assign to minimum value in a column. Default color is red.
highlighter	color to assign the background of a cell containing minimum value in a column.

Value

a function that applies a color to the minimum value in a column of numeric values.

Examples

```
data <- MASS::road[11:17, ]

## By default, the minimum value is bold with a red font color
reactable(data,
defaultColDef = colDef(
    style = highlight_min(data)))

## Assign a different font color
reactable(data,
defaultColDef = colDef(
    style = highlight_min(data,
    font_color = "green")))

## Highlight the background of the cell for the minimum value in each column
reactable(data,
defaultColDef = colDef(
    style = highlight_min(data,
    highlighter = "yellow")))
```

`highlight_min_max` *Highlights the minimum and maximum value in a column*

Description

The 'highlight_min_max()' function assigns a font color and/or background color to both the minimum and maximum values in a column. It should be placed within the style argument in reactable::colDef.

Usage

```
highlight_min_max(
  data,
  min_font_color = "red",
  max_font_color = "green",
  min_highlighter = NULL,
  max_highlighter = NULL
)
```

Arguments

`data`	Dataset containing at least one numeric column.
`min_font_color`	color to assign to minimum value in a column. Default color is red.
`max_font_color`	color to assign to maximum value in a column. Default color is green.
`min_highlighter`	color to assign the background of a cell containing minimum value in a column.
`max_highlighter`	color to assign the background of a cell containing maximum value in a column.

Value

a function that applies a color to the minimum and maximum values in a column of numeric values.

Examples

```
data <- MASS::road[11:17, ]

## By default, the minimum and maximum values are bold with a red and green font color respectively
reactable(data,
defaultColDef = colDef(
    style = highlight_min_max(data)))

## Assign a different font color to the min and max values
reactable(data,
defaultColDef = colDef(
    style = highlight_min_max(data,
    min_font_color = "orange",
    max_font_color = "blue")))

## Highlight the background of the cell for the min and max values in each column
reactable(data,
defaultColDef = colDef(
    style = highlight_min_max(data,
    min_highlighter = "salmon",
    max_highlighter = "skyblue")))
```

---

icon_assign *Assign icons to rows in a column*

Description

The ‘icon_assign()‘ function assigns icons from the Font Awesome library (via shiny) to each row of a numeric column depending on the value in each row. By default, the number of icons assigned will be equal to the value in that row. If the value is less than the max, it will receive empty icons. Both the icon shape and color of the filled and empty icons can be modified through the parameters. It should be placed within the cell argument in reactable::colDef.

Usage

```
icon_assign(
  data,
  icon = "circle",
  fill_color = "#1e90ff",
  empty_color = "lightgrey",
  buckets = NULL,
  number_fmt = NULL,
  seq_by = 1,
  show_values = FALSE
)
```

Arguments

data	Dataset containing at least one numeric column.
icon	A single icon from the Font Awesome library (via shiny). Default icon is a circle.
fill_color	A single color for the filled icons. Default color is #1e90ff.
empty_color	A single color for the empty icons. Default color is lightgrey.
buckets	Optionally divide values in a column into buckets by providing a numeric value. Icons are then assigned by rank from lowest to highest. Default is set to NULL.
number_fmt	Optionally format numbers using formats from the scales package. Default is set to NULL.
seq_by	A numerical input that determines what number each icon represents. Ex. instead of displaying 100 icons for the number 100, can set seq_by = 10 to show only 10 icons. Default value is set to 1.
show_values	Optionally display values next to icons. Icons can be displayed to the left of the icons with "left" or to the right with "right". Default is set to FALSE or no values.

Value

a function that applies colored icons to a column of numeric values.

Examples

```
data <- iris[10:29, ]
## By default, icon_assign() assigns a cirlce icon for each value up to the maximum value.
## If a value is 5 and the maximum value in the column is 6,
## It will assign 5 blue icons and 1 grey icon.
reactable(data,
columns = list(
Sepal.Length = colDef(cell = icon_assign(data))))

## Assign colors to filled icons and empty icons
reactable(data,
columns = list(
Sepal.Length = colDef(cell = icon_assign(data,
fill_color = "red",
empty_color = "white"))))

## Assign any icon from the Font Awesome Library
reactable(data,
columns = list(
Sepal.Length = colDef(cell = icon_assign(data,
icon = "fan"))))

## Optionally divide values into buckets and assign icons based on rank.
reactable(data,
columns = list(
Sepal.Length = colDef(cell = icon_assign(data,
```

```
buckets = 3))))

## Optionally display values next to icons.
reactable(data,
columns = list(
Sepal.Length = colDef(cell = icon_assign(data,
show_values = "right"))))
```

---

icon_sets *Add colored icons to rows in a column*

Description

The 'icon_sets()' function conditionally adds an icon from the Font Awesome library (via shiny) to each row of a column and assigns a color depending on their value in relation to other values in that particular column. It should be placed within the cell argument in reactable::colDef.

Usage

```
icon_sets(
  data,
  icons = c("circle", "circle", "circle"),
  colors = c("red", "orange", "green"),
  number_fmt = NULL
)
```

Arguments

data	Dataset containing at least one numeric column.
icons	A vector of three icons from the Font Awesome library (via shiny). Icons should be given in order from low values to high values. Default icons are circles.
colors	A vector of three colors to color the icons. Colors should be given in order from low values to high values. Default colors provided are c('red','orange','green'). Can use R's built-in colors or other color packages.
number_fmt	Optionally format numbers using formats from the scales package. Default is set to NULL.

Value

a function that applies an icon to a column of numeric values.

Examples

```
data <- MASS::Cars93[20:49, c("Make", "MPG.city", "MPG.highway")]

## By default, icon_sets() assigns red circles to the lowest-third values,
## orange circles to the middle-third values,
## and green to the top-third values
reactable(data,
defaultColDef = colDef(cell = icon_sets(data)))

## Assign custom colors
reactable(data,
defaultColDef = colDef(cell = icon_sets(data,
colors = c("tomato", "grey", "dodgerblue"))))

## Assign icons from Font Awesome's icon library
reactable(data,
defaultColDef = colDef(cell = icon_sets(data,
icons = c("arrow-down","minus","arrow-up"))))

## Use number_fmt to format numbers using the scales package
car_prices <- MASS::Cars93[20:49, c("Make", "Price")]

reactable(car_prices,
defaultColDef = colDef(cell = icon_sets(car_prices,
number_fmt = scales::dollar)))
```

---

pos_neg_colors *Assign colors to negative and positive values*

Description

The 'pos_neg_colors()' function assigns a color to all negative values and a color to all positive values. It should be placed within the style argument in reactable::colDef.

Usage

```
pos_neg_colors(neg_col, pos_col, bold = NULL)
```

Arguments

neg_col	color to assign to negative values.
pos_col	color to assign to positive values.
bold	optional argument to bold values. Default is set to NULL or not bold.

Value

a function that applies a color to the positive and negative values of numeric column.

Examples

```
data <- data.frame(
Symbol = c("GOOG", "FB", "AMZN", "NFLX", "TSLA"),
Price = c(1265.13, 187.89, 1761.33, 276.82, 328.13),
Change = c(4.14, 1.51, -19.45, 5.32, -12.45))

## Assign the color red to negative values and green to positive values
reactable(data,
columns = list(
Change = colDef(
style = pos_neg_colors("red", "green"))))

## Bold values
reactable(data,
columns = list(
Change = colDef(
style = pos_neg_colors("red", "green", bold = TRUE))))
```

Index