# Package ‘reactablefmtr’

June 16, 2021

**Type** Package

**Title** Easily Customize Interactive Tables Made with Reactable

**Version** 1.0.0

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

**Description** Enhance the styling of interactive reactable tables with easy-to-use and highly-customizable functions. Apply conditional formatting to cells with data bars, color scales, and icon sets. Utilize custom table themes inspired by popular websites and bootstrap themes. Increase the portability and reproducibility of reactable tables by embedding images from the web directly into cells. Save the final table output as a static image or interactive file (note this feature requires the 'webshot2' package which can be downloaded from <https://github.com/rstudio/webshot2>).

**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

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

**Imports** dplyr, grDevices, htmltools, htmlwidgets, shiny, stringr, tools

**Suggests** MASS, scales, webshot2

**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** 2021-06-16 04:10:05 UTC

## R topics documented:

`add_source` *Add a source below a reactable table*

## Description

Use ‘add_source()‘ to place a source below a reactable or reactablefmtr table. The same options that are present in ‘add_title()‘ and ‘add_subtitle()‘ are also available in ‘add_source()‘. The source can be aligned to the left, right, or center with the align option. The text properties of the source, such as the font size, font family, and font style can be customized. The background color of the source can also be adjusted as well as the margin around the source.

## Usage

```
add_source(
  table = NULL,
  source = NULL,
  align = "left",
  font_color = "#000",
  font_family = "-apple-system,BlinkMacSystemFont,Helvetica,Arial,sans-serif",
  font_size = 16,
  font_style = "normal",
  font_weight = "normal",
  text_decoration = NULL,
  background_color = "#FFFFFF",
  margin = 4
)
```

## Arguments

| | |
|---|---|
| `table` | A reactable table. |
| `source` | A string to be displayed as the source. |
| `align` | The alignment of the source. Options are "left", "right", "center". Default is "left". |
| `font_color` | Color of the source text. Default is #000. |
| `font_family` | Font family of the source. Default is -apple-system, BlinkMacSystemFont, Helvetica, Arial, sans-serif. |
| `font_size` | Numeric value representing the size of the font of the source (in px). Default is 16. |
| `font_style` | Style of the source font. Options are "normal" or "italic". Default is "normal". |
| `font_weight` | The font weight of the source. Options are "bold" or "normal". Default is "normal". |
| `text_decoration` | Optionally add an underline, overline, or line-through source. Options are "underline", "overline", "underline overline", or "line-through". Default is NULL. |

| | |
|---|---|
| `background_color` | Color of the source background. Default is #FFFFFF. |
| `margin` | Numeric value representing the four-sided margin around the source (in px). Default is 4. |

## Value

a function that adds a source below a reactable table.

## Examples

```
## Not run:
## Create the reactable table and then pipe in the source
table <- reactable(iris[10:29, ])

table %>%
  add_source("This is a source")

## Use options to adjust the style and position of the source
table %>%
  add_source("This is a source", font_style = "italic", font_color = "grey")

## End(Not run)
```

---

`add_subtitle` *Add a subtitle above a reactable table*

---

## Description

Use ‘add_subtitle()‘ to place a subtitle above a reactable or reactablefmtr table. The same options that are present in ‘add_title()‘ and ‘add_source()‘ are also available in ‘add_subtitle()‘. The subtitle can be aligned to the left, right, or center with the align option. The text properties of the subtitle, such as the font size, font family, and font style can be customized. The background color of the subtitle can also be adjusted as well as the margin around the subtitle.

## Usage

```
add_subtitle(
  table = NULL,
  subtitle = NULL,
  align = "left",
  font_color = "#333",
  font_family = "-apple-system,BlinkMacSystemFont,Helvetica,Arial,sans-serif",
  font_size = 24,
  font_style = "normal",
  font_weight = "bold",
  text_decoration = NULL,
  background_color = "#FFFFFF",
  margin = 2
)
```

## Arguments

| | |
|---|---|
| `table` | A reactable table. |
| `subtitle` | A string to be displayed as the subtitle. |
| `align` | The alignment of the subtitle. Options are "left", "right", "center". Default is "left". |
| `font_color` | Color of the subtitle text. Default is #333. |
| `font_family` | Font family of the subtitle. Default is -apple-system, BlinkMacSystemFont, Helvetica, Arial, sans-serif. |
| `font_size` | Numeric value representing the size of the font of the subtitle (in px). Default is 24. |
| `font_style` | Style of the subtitle font. Options are "normal" or "italic". Default is "normal". |
| `font_weight` | The font weight of the subtitle. Options are "bold" or "normal". Default is "bold". |
| `text_decoration` | Optionally add an underline, overline, or line-through subtitle. Options are "underline", "overline", "underline overline", or "line-through". Default is NULL. |
| `background_color` | Color of the subtitle background. Default is #FFFFFF. |
| `margin` | Numeric value representing the four-sided margin around the subtitle (in px). Default is 2. |

## Value

a function that adds a subtitle above a reactable table.

## Examples

```
## Not run:
## Create the reactable table and then pipe in the subtitle
table <- reactable(iris[10:29, ])

table %>%
  add_subtitle("This is a subtitle")

## If a title proceeds a subtitle, the subtite will be placed below the title
table %>%
  add_title("This is a title") %>%
  add_subtitle("This is a subtitle")

## Use options to adjust the style and position of the subtitle
table %>%
  add_subtitle("This is a subtitle", align = "center", font_color = "red")

## End(Not run)
```

`add_title` *Add a title above a reactable table*

## Description

Use ‘add_title()‘ to place a title above a reactable or reactablefmtr table. The title can be aligned to the left, right, or center with the align option. The text properties of the title, such as the font size, font family, and font style can be customized. The background color of the title can also be adjusted as well as the margin around the title.

## Usage

```
add_title(
  table = NULL,
  title = NULL,
  align = "left",
  font_color = "#000",
  font_family = "-apple-system,BlinkMacSystemFont,Helvetica,Arial,sans-serif",
  font_size = 32,
  font_style = "normal",
  font_weight = "bold",
  text_decoration = NULL,
  background_color = "#FFFFFF",
  margin = 2
)
```

## Arguments

| | |
|---|---|
| `table` | A reactable table. |
| `title` | A string to be displayed as the title. |
| `align` | The alignment of the table. Options are "left", "right", "center". Default is "left". |
| `font_color` | Color of the title text. Default is #000. |
| `font_family` | Font family of the title. Default is -apple-system, BlinkMacSystemFont, Helvetica, Arial, sans-serif. |
| `font_size` | Numeric value representing the size of the font of the title (in px). Default is 32. |
| `font_style` | Style of the title font. Options are "normal" or "italic". Default is "normal". |
| `font_weight` | The font weight of the title. Options are "bold" or "normal". Default is "bold". |
| `text_decoration` | Optionally add an underline, overline, or line-through title. Options are "underline", "overline", "underline overline", or "line-through". Default is NULL. |
| `background_color` | Color of the title background. Default is #FFFFFF. |
| `margin` | Numeric value representing the four-sided margin around the title (in px). Default is 2. |

## Value

a function that adds a title above a reactable table.

## Examples

```
## Not run:
## Create the reactable table and then pipe in the title
table <- reactable(iris[10:29, ])

table %>%
  add_title("This is a title")

## Use options to adjust the style and position of the title
table %>%
  add_title("This is a title", align = "center", font_color = "red")

## End(Not run)
```

---

`cerulean` *Theme cerulean*

---

## Description

Bootstrap-inspired cerulean theme

## Usage

```
cerulean(
  font_family = "Verdana",
  font_size = 14,
  font_color = "#141415",
  header_font_family = "Verdana",
  header_font_size = 15,
  header_font_color = "#cfe9f7",
  cell_padding = 6
)
```

## Arguments

| | |
|---|---|
| `font_family` | Font family for the text within the table. Default is Verdana. |
| `font_size` | Numeric value representing the size of the font within the table (in px). Default is 14. |
| `font_color` | Color of the font for the text within the table and the group headers. Default is #141415. |
| `header_font_family` | Font family for the header text. Default is Verdana. |

| | |
|---|---|
| `header_font_size` | Numeric value representing the size of the font within the table (in px). Default is 15. |
| `header_font_color` | Color of the font for the header text. Default is #cfe9f7. |
| `cell_padding` | Numeric value representing the padding size between cells (in px). Default is 6. |

## Value

an object of class theme that is applied to a reactable table.

## Examples

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

## Standard cerulean theme
reactable(data,
          theme = cerulean())

## Cerulean theme with additional options applied
reactable(data,
          theme = cerulean(font_size = 12, font_color = "grey", cell_padding = 3))
```

---

`clean` *Theme clean*

---

## Description

Simple clean-look theme

## Usage

```
clean(
  font_family = "Verdana",
  font_size = 14,
  font_color = "#222222",
  header_font_family = "Verdana",
  header_font_size = 15,
  header_font_color = "#222222",
  cell_padding = 6
)
```

## Arguments

| | |
|---|---|
| `font_family` | Font family for the text within the table. Default is Verdana. |
| `font_size` | Numeric value representing the size of the font within the table (in px). Default is 14. |

| | |
|---|---|
| `font_color` | Color of the font for the text within the table. Default is #222222. |
| `header_font_family` | Font family for the header text. Default is Verdana. |
| `header_font_size` | Numeric value representing the size of the font within the table (in px). Default is 15. |
| `header_font_color` | Color of the font for the header text. Default is #222222. |
| `cell_padding` | Numeric value representing the padding size between cells (in px). Default is 6. |

## Value

an object of class theme that is applied to a reactable table.

## Examples

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

## Standard clean theme
reactable(data,
          theme = clean())

## Cerulean theme with additional options applied
reactable(data,
          theme = clean(font_size = 12, font_color = "grey", cell_padding = 3))
```

---

## `color_scales` *Add color scales to cells 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. The colors can be provided within a vector in ‘colors‘ or via another column in the dataset by referencing the column by name with ‘color_ref‘. The opacity of the colors provided can be adjusted by providing a value between 0 and 1 in ‘opacity‘. ‘text_color‘ can be used to change the color of the values. If values are displayed within a dark-colored background, ‘brighten_text‘ will display the values in white text so they are more visible. The color of ‘brighten_text_color‘ can be changed to a color other than white if desired. If the user wants to assign colors row-wise instead of column-wise, set ‘span‘ equal to TRUE to apply across all columns. Or can provide the names of the columns by either column name or column position number to apply to only a subset of the columns. ‘color_scales()‘ should be placed within the style argument in reactable::colDef.

## Usage

```
color_scales(
  data,
  colors = c("#67a9cf", "#f8fcf8", "#ef8a62"),
  color_ref = NULL,
  opacity = 1,
  text_color = "black",
  show_text = TRUE,
  brighten_text = TRUE,
  brighten_text_color = "white",
  bold_text = FALSE,
  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 blue-white-orange: c("#67a9cf", "#f8fcf8", "#ef8a62"). Can use R's built-in colors or other color packages. |
| `color_ref` | Optionally assign colors to from another column by providing the name of the column containing the colors in quotes. Only one color can be provided per row. Default is NULL. |
| `opacity` | A value between 0 and 1 that adjusts the opacity in colors. A value of 0 is fully transparent, a value of 1 is fully opaque. Default is 1. |
| `text_color` | Assigns text color to values. Default is black. |
| `show_text` | Logical: show text or hide text. Default is TRUE. |
| `brighten_text` | Logical: automatically assign color to text based on background color of cell. Text within dark-colored backgrounds will turn white, text within light-colored backgrounds will be black. Default is TRUE. |
| `brighten_text_color` | Assigns text color to values if values are within a dark-colored backgrounds. Default is white. |
| `bold_text` | Logical: bold text. Default is 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 blue-white-orange 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 cells 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 colors can be provided within a vector in ‘colors‘ or via another column in the dataset by referencing the column by name with ‘color_ref‘. The opacity of the colors provided can be adjusted by providing a value between 0 and 1 in ‘opacity‘. ‘text_color‘ can be used to change the color of the values. If values are displayed within a dark-colored background, ‘brighten_text‘ will display the values in white text so they are more visible. The color of ‘brighten_text_color‘ can be changed to a color other than white if desired. If the user wants to assign colors row-wise instead of column-wise, set ‘span‘ equal to TRUE to apply across all columns. Or can provide the names of the columns by either column name or column position number to apply to only a subset of the columns. ‘color_tiles()‘ needs to placed within the cell argument in reactable::colDef.

## Usage

```
color_tiles(
  data,
  colors = c("#67a9cf", "#f8fcf8", "#ef8a62"),
  color_ref = NULL,
  opacity = 1,
  number_fmt = NULL,
  text_color = "black",
  show_text = TRUE,
  brighten_text = TRUE,
  brighten_text_color = "white",
  bold_text = FALSE,
  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 blue-white-orange: c("#67a9cf", "#f8fcf8", "#ef8a62"). Can use R's built-in colors or other color packages. |
| `color_ref` | Optionally assign colors to from another column by providing the name of the column containing the colors in quotes. Only one color can be provided per row. Default is NULL. |
| `opacity` | A value between 0 and 1 that adjusts the opacity in colors. A value of 0 is fully transparent, a value of 1 is fully opaque. Default is 1. |
| `number_fmt` | Optionally format numbers using formats from the scales package. Default is set to NULL. |
| `text_color` | Assigns text color to values. Default is black. |
| `show_text` | Logical: show text or hide text. Default is TRUE. |
| `brighten_text` | Logical: automatically assign color to text based on background color of cell. Text within dark-colored backgrounds will turn white, text within light-colored backgrounds will be black. Default is TRUE. |
| `brighten_text_color` | Assigns text color to values if values are within a dark-colored backgrounds. Default is white. |
| `bold_text` | Logical: bold text. Default is 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 blue-white-orange 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)))
```

---

`cosmo` *Theme cosmo*

---

## Description

Bootstrap-inspired cosmo theme

## Usage

```
cosmo(
  font_family = "Verdana",
  font_size = 14,
  font_color = "#141415",
  header_font_family = "Verdana",
  header_font_size = 15,
  header_font_color = "#ffffff",
  cell_padding = 6
)
```

## Arguments

| | |
|---|---|
| font_family | Font family for the text within the table. Default is Verdana. |
| font_size | Numeric value representing the size of the font within the table (in px). Default is 14. |
| font_color | Color of the font for the text within the table and the group headers. Default is #141415. |
| header_font_family | Font family for the header text. Default is Verdana. |
| header_font_size | Numeric value representing the size of the font within the table (in px). Default is 15. |
| header_font_color | Color of the font for the header text. Default is #ffffff. |
| cell_padding | Numeric value representing the padding size between cells (in px). Default is 6. |

## Value

an object of class theme that is applied to a reactable table.

## Examples

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

## Standard cosmo theme
reactable(data,
          theme = cosmo())

## Cerulean theme with additional options applied
reactable(data,
          theme = cosmo(font_size = 12, font_color = "grey", cell_padding = 3))
```

---

cyborg *Theme cyborg*

---

## Description

Bootstrap-inspired cyborg theme

## Usage

```
cyborg(
  font_family = "Verdana",
  font_size = 14,
  font_color = "#888888",
  header_font_family = "Verdana",
  header_font_size = 15,
  header_font_color = "#7b7b7b",
  cell_padding = 6
)
```

## Arguments

| | |
|---|---|
| `font_family` | Font family for the text within the table. Default is Verdana. |
| `font_size` | Numeric value representing the size of the font within the table (in px). Default is 14. |
| `font_color` | Color of the font for the text within the table and the group headers. Default is #888888. |
| `header_font_family` | Font family for the header text. Default is Verdana. |
| `header_font_size` | Numeric value representing the size of the font within the table (in px). Default is 15. |
| `header_font_color` | Color of the font for the header text. Default is #7b7b7b. |
| `cell_padding` | Numeric value representing the padding size between cells (in px). Default is 6. |

## Value

an object of class theme that is applied to a reactable table.

## Examples

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

## Standard cyborg theme
reactable(data,
          theme = cyborg())

## Cerulean theme with additional options applied
reactable(data,
          theme = cyborg(font_size = 12, font_color = "grey", cell_padding = 3))
```

`darkly` *Theme darkly*

## Description

Bootstrap-inspired darkly theme

## Usage

```
darkly(
  font_family = "Georgia",
  font_size = 14,
  font_color = "#ffffff",
  header_font_family = "Georgia",
  header_font_size = 15,
  header_font_color = "#afbdcc",
  cell_padding = 6
)
```

## Arguments

| | |
|---|---|
| `font_family` | Font family for the text within the table. Default is Georgia. |
| `font_size` | Numeric value representing the size of the font within the table (in px). Default is 14. |
| `font_color` | Color of the font for the text within the table and the group headers. Default is #ffffff. |
| `header_font_family` | Font family for the header text. Default is Georgia. |
| `header_font_size` | Numeric value representing the size of the font within the table (in px). Default is 15. |
| `header_font_color` | Color of the font for the header text. Default is #afbdcc. |
| `cell_padding` | Numeric value representing the padding size between cells (in px). Default is 6. |

## Value

an object of class theme that is applied to a reactable table.

## Examples

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

## Standard darkly theme
reactable(data,
          theme = darkly())
```

```
## Cerulean theme with additional options applied
reactable(data,
          theme = darkly(font_size = 12, font_color = "grey", cell_padding = 3))
```

---

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

---

## Description

The ‘data_bars()‘ function 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. The maximum width of the filled bars can be adjusted. Ex. if you are displaying percentages, and the maximum value in your column is 50 you could increase the maximum fill to 100 The values for the bars can be displayed inside or outside the filled bars with the ‘text_position‘ option. By default, the values are displayed on the outside-end of the filled bars. The fill_color of both the fill and the background of the bars can be adjusted. To adjust the fill_color of the filled bar, use ‘fill_color‘. If more than one color is provided, a conditional color palette will be applied to to the values, or if ‘fill_gradient‘ is set to TRUE, a left-to-right gradient fill color will be applied. The fill colors can also be provided via another column in the dataset by referencing the column by name with ‘fill_color_ref‘. ‘text_color‘ can be used to change the color of the text_position. By default, the label color is black. If values are displayed inside the bars and a dark color palette is used to fill the bars, ‘brighten_text‘ will display the values in white text so the values are visible by default. The color of ‘brighten_text_color‘ can be changed to a color other than white if desired. An icon or image can be added to the data bars with ‘icon‘ or ‘img‘. Alternatively, icons and images can be assigned from another column with ‘icon_ref‘ and ‘img_ref‘, similar to ‘fill_color_ref‘. The color of the icons can be assigned through either ‘icon_color‘ (a single color) or ‘icon_color_ref‘ (from another column). The size of the images can be adjusted using ‘img_height‘ and ‘img_width‘. The size of the icons can be adjusted using ‘icon_size‘. ‘data_bars()‘ works with columns containing both positive and negative values. It should be placed within the cell argument in reactable::colDef.

## Usage

```
data_bars(
  data,
  fill_color = "#1e90ff",
  fill_color_ref = NULL,
  fill_opacity = 1,
  fill_gradient = FALSE,
  background = "transparent",
  max_value = NULL,
  min_value = NULL,
  align_bars = "left",
  bar_height = 19,
  text_position = "outside-end",
  text_color = "black",
  brighten_text = TRUE,
  brighten_text_color = "white",
  bold_text = FALSE,
  number_fmt = NULL,
  icon = NULL,
  icon_ref = NULL,
  icon_size = 20,
  icon_color = NULL,
  icon_color_ref = NULL,
  img = NULL,
  img_ref = NULL,
  img_height = 20,
  img_width = 20
)
```

## Arguments

| | |
|---|---|
| `data` | Dataset containing at least one numeric column. |
| `fill_color` | A single color or a vector of fill_color for the fill of the data bars. fill_color should be given in order from low values to high values. Can use R's built-in fill_color or other color packages. Default is #1e90ff. |
| `fill_color_ref` | Optionally assign fill_color to from another column by providing the name of the column containing the fill colors in quotes. Only one color can be provided per row, and therefore will not work with fill_gradient. Default is NULL. |
| `fill_opacity` | A value between 0 and 1 that adjusts the opacity in fill_color. A value of 0 is fully transparent, a value of 1 is fully opaque. Default is 1. |
| `fill_gradient` | Logical: if two or more colors are provided in fill_color, the colors in the fill of the bars are converted to a left-to-right gradient. Default is FALSE. |
| `background` | The color for the background of the data bars. Default is transparent. |
| `max_value` | A value to use as the maximum value for the width of the filled bars. The default maximum value is the maximum value in the column. Default is NULL. |
| `min_value` | A value to use as the minimum value for the width of the filled bars. Default is NULL. |
| `align_bars` | Display filled bars from left-to-right or right-to-left. Options are "left" or "right". Default is left. |
| `bar_height` | Numeric height of the data bars in px. Default is 19. |
| `text_position` | Choose where to display the values. Values can be displayed within the filled bars ("inside-end" or "inside-base"), outside of the filled bars ("outside-end" or "outside-base"), within the center of the filled bars ("center"), or not displayed at all ("none"). Default is outside-end. |
| `text_color` | Assigns text color to values. Default is black. |
| `brighten_text` | Logical: automatically assign color to text based on filled color when the text is positioned within the filled bars. Text within dark-colored filled bars will turn white, text within light-colored bars will be black. Default is TRUE. |
| `brighten_text_color` | Assigns text color to values if values are within a dark-colored filled bar. Default is white. |

| | |
|---|---|
| `bold_text` | Logical: bold text. Default is FALSE. |
| `number_fmt` | Optionally format numbers using formats from the scales package. Default is NULL. |
| `icon` | An icon from the Font Awesome library (via shiny). If an icon is provided, it will be positioned so that it does not overlap the text for the data bars. Default is NULL. |
| `icon_ref` | Optionally assign icons from another column by providing the name of the column containing the icons in quotes. Only one icon can be provided per cell. Default is NULL. |
| `icon_size` | A value representing the size of the icon in px. Default is 20. |
| `icon_color` | The color for the icon. If no color is provided, default is set to the color of the filled bars. Default is NULL. |
| `icon_color_ref` | Optionally assign color to the icons from another column by providing the name of the column containing the icon colors in quotes. Only one color can be provided per cell. Default is NULL. |
| `img` | An image provided with a valid URL. |
| `img_ref` | Optionally assign images from another column by providing the name of the column containing the image URLs in quotes. Only one image can be provided per cell. Default is NULL. |
| `img_height` | A value for the height of the image in px. Default is 20. |
| `img_width` | A value for the width of the image in px. Default is 20. |

## Value

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

## Examples

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

## By default, data bars are aligned left and text_position are placed on the outside end
reactable(data,
defaultColDef = colDef(
cell = data_bars(data)))

## Align the bars to the right
reactable(data,
defaultColDef = colDef(
cell = data_bars(data,
align = "right")))

## Move the text values inside the filled bars
reactable(data,
defaultColDef = colDef(
cell = data_bars(data,
text_position = "inside-end")))

## Apply multiple fill_color to the filled bars
```

```
reactable(data,
defaultColDef = colDef(
cell = data_bars(data,
fill_color = c("lightblue","royalblue","navy"))))

## Apply a fill_gradient pattern to the filled bars
reactable(data,
defaultColDef = colDef(
cell = data_bars(data,
fill_color = c("lightblue","royalblue","navy"),
fill_gradient = TRUE)))
```

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

### Description

The ‘data_bars_gradient()‘ function is depreciated. The new version of ‘data_bars()‘ can convert colors into gradients with ‘gradient = TRUE‘. Please use ‘data_bars()‘ instead.

### 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

```
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(data,
fill_color = c("#1efffd", "#1e20ff"),
fill_gradient = TRUE)))
```

---

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

---

### Description

The ‘data_bars_pos_neg()‘ function is depreciated. The new version of ‘data_bars()‘ can handle both positive and negative values now. Please use ‘data_bars()‘ instead.

### 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(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(data,
  fill_color = c("#ff3030", "#ffffff", "#1e90ff")))))
```

---

default *Theme default*

---

## Description

Reactable-inspired default theme

## Usage

```
default(
  font_family = "-apple-system,BlinkMacSystemFont,Helvetica,Arial,sans-serif",
  font_size = 15,
  font_color = "#333333",
  header_font_family = "-apple-system,BlinkMacSystemFont,Helvetica,Arial,sans-serif",
  header_font_size = 15,
  header_font_color = "#333333",
  cell_padding = 6
)
```

## Arguments

| | |
|---|---|
| `font_family` | Font family for the text within the table. Default is -apple-system, BlinkMac-SystemFont, Helvetica, Arial, sans-serif. |

| | |
|---|---|
| `font_size` | Numeric value representing the size of the font within the table (in px). Default is 15. |
| `font_color` | Color of the font for the text within the table and the group headers. Default is #333333. |
| `header_font_family` | Font family for the header text. Default is -apple-system, BlinkMacSystemFont, Helvetica, Arial, sans-serif. |
| `header_font_size` | Numeric value representing the size of the font within the table (in px). Default is 15. |
| `header_font_color` | Color of the font for the header text. Default is #333333. |
| `cell_padding` | Numeric value representing the padding size between cells (in px). Default is 6. |

## Value

an object of class theme that is applied to a reactable table.

## Examples

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

## Standard default theme
reactable(data,
         theme = default())

## Default theme with additional options applied
reactable(data,
         theme = default(font_size = 12, font_color = "grey", cell_padding = 3))
```

---

`embed_img` *Embed image from web to cells 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,
  label_position = "right"
)
```

## 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. |
| `label_position` | Position of label relative to image. Options are "right", "left", "below", or "above". Default is right. |

## 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"))))
```

## espn *Theme espn*

### Description

ESPN-inspired table theme

### Usage

```
espn(
  font_family = "Arial",
  font_size = 12,
  font_color = "#6C6D6F",
  header_font_family = "Arial",
  header_font_size = 11,
  header_font_color = "#48494a",
  cell_padding = 7
)
```

### Arguments

| | |
|---|---|
| `font_family` | Font family for the text within the table. Default is Arial. |
| `font_size` | Numeric value representing the size of the font within the table (in px). Default is 12. |
| `font_color` | Color of the font for the text within the table and the group headers. Default is #6C6D6F. |
| `header_font_family` | Font family for the header text. Default is Arial. |
| `header_font_size` | Numeric value representing the size of the font within the table (in px). Default is 11. |
| `header_font_color` | Color of the font for the header text. Default is #48494a. |
| `cell_padding` | Numeric value representing the padding size between cells (in px). Default is 7. |

### Value

an object of class theme that is applied to a reactable table.

### Examples

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

## Standard espn theme
reactable(data,
          theme = espn())
```

```
## Cerulean theme with additional options applied
reactable(data,
          theme = espn(font_size = 12, font_color = "grey", cell_padding = 3))
```

## `fivethirtyeight` *Theme fivethirtyeight*

### Description

538-inspired table theme

### Usage

```
fivethirtyeight(
  font_family = "Helvetica",
  font_size = 14,
  font_color = "#222222",
  header_font_family = "Helvetica",
  header_font_size = 12,
  header_font_color = "#000000",
  cell_padding = 5
)
```

### Arguments

| | |
|---|---|
| `font_family` | Font family for the text within the table. Default is Helvetica. |
| `font_size` | Numeric value representing the size of the font within the table (in px). Default is 14. |
| `font_color` | Color of the font for the text within the table and the group headers. Default is #222222. |
| `header_font_family` | Font family for the header text. Default is Helvetica. |
| `header_font_size` | Numeric value representing the size of the font within the table (in px). Default is 12. |
| `header_font_color` | Color of the font for the header text. Default is #000000. |
| `cell_padding` | Numeric value representing the padding size between cells (in px). Default is 5. |

### Value

an object of class theme that is applied to a reactable table.

## Examples

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

## Standard fivethirtyeight theme
reactable(data,
          theme = fivethirtyeight())

## Cerulean theme with additional options applied
reactable(data,
         theme = fivethirtyeight(font_size = 12, font_color = "grey", cell_padding = 3))
```

---

`flatly` *Theme flatly*

---

## Description

Bootstrap-inspired flatly theme

## Usage

```
flatly(
  font_family = "Georgia",
  font_size = 14,
  font_color = "#212529",
  header_font_family = "Georgia",
  header_font_size = 15,
  header_font_color = "#ffffff",
  cell_padding = 6
)
```

## Arguments

| | |
|---|---|
| `font_family` | Font family for the text within the table. Default is Georgia. |
| `font_size` | Numeric value representing the size of the font within the table (in px). Default is 14. |
| `font_color` | Color of the font for the text within the table and the group headers. Default is #212529. |
| `header_font_family` | Font family for the header text. Default is Georgia. |
| `header_font_size` | Numeric value representing the size of the font within the table (in px). Default is 15. |
| `header_font_color` | Color of the font for the header text. Default is #ffffff. |
| `cell_padding` | Numeric value representing the padding size between cells (in px). Default is 6. |

## Value

an object of class theme that is applied to a reactable table.

## Examples

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

## Standard flatly theme
reactable(data,
          theme = flatly())

## Cerulean theme with additional options applied
reactable(data,
          theme = flatly(font_size = 12, font_color = "grey", cell_padding = 3))
```

---

## 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")))
```

## hoverdark *Theme hoverdark*

### Description

Changes from light-themed to dark-themed on hover

### Usage

```
hoverdark(
  font_family = "Verdana",
  font_size = 15,
  font_color = "#222222",
  header_font_family = "Verdana",
  header_font_size = 15,
  cell_padding = 4
)
```

### Arguments

| | |
|---|---|
| `font_family` | Font family for the text within the table. Default is Verdana. |
| `font_size` | Numeric value representing the size of the font within the table (in px). Default is 15. |
| `font_color` | Color of the font for the text within the table. Default is #222222. |
| `header_font_family` | Font family for the header text. Default is Verdana. |
| `header_font_size` | Numeric value representing the size of the font within the table (in px). Default is 15. |
| `cell_padding` | Numeric value representing the padding size between cells (in px). Default is 4. |

### Value

an object of class theme that is applied to a reactable table.

### Examples

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

## Standard hoverdark theme
reactable(data,
          theme = hoverdark())

## Cerulean theme with additional options applied
reactable(data,
          theme = hoverdark(font_size = 12, font_color = "grey", cell_padding = 3))
```

---

## hoverlight *Theme hoverlight*

---

### Description

Changes from dark-themed to light-themed on hover

### Usage

```
hoverlight(
  font_family = "Verdana",
  font_size = 15,
  font_color = "#ffffff",
  header_font_family = "Verdana",
  header_font_size = 15,
  cell_padding = 4
)
```

### Arguments

| | |
|---|---|
| `font_family` | Font family for the text within the table. Default is Verdana. |
| `font_size` | Numeric value representing the size of the font within the table (in px). Default is 15. |
| `font_color` | Color of the font for the text within the table. Default is #ffffff. |
| `header_font_family` | Font family for the header text. Default is Verdana. |
| `header_font_size` | Numeric value representing the size of the font within the table (in px). Default is 15. |
| `cell_padding` | Numeric value representing the padding size between cells (in px). Default is 4. |

## Value

an object of class theme that is applied to a reactable table.

## Examples

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

## Standard hoverlight theme
reactable(data,
          theme = hoverlight())

## Cerulean theme with additional options applied
reactable(data,
          theme = hoverlight(font_size = 12, font_color = "grey", cell_padding = 3))
```

---

# `icon_assign` *Assign icons to cells in a column*

---

## Description

The 'icon_assign()' function assigns icons from the Font Awesome library (via shiny) to each cell 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 cell. If the value is less than the max, it will receive empty icons. Both the icon shape, size, and color of the filled and empty icons can be modified through the parameters. Values can optionally be shown with the icons if desired. It should be placed within the cell argument in reactable::colDef.

## Usage

```
icon_assign(
  data,
  icon = "circle",
  fill_color = "#1e90ff",
  empty_color = "lightgrey",
  fill_opacity = 1,
  empty_opacity = 1,
  icon_size = 16,
  buckets = NULL,
  number_fmt = NULL,
  seq_by = 1,
  show_values = "none"
)
```

## 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. |
| `fill_opacity` | A value between 0 and 1 that adjusts the opacity in fill_color. A value of 0 is fully transparent, a value of 1 is fully opaque. Default is 1. |
| `empty_opacity` | A value between 0 and 1 that adjusts the opacity in empty_color. A value of 0 is fully transparent, a value of 1 is fully opaque. Default is 1. |
| `icon_size` | A value representing the size of the icon in px. Default is 16. |
| `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. Options are "left", "right", above", "below", or "none". Default is none. |

## 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 cells in a column*

---

## Description

The ‘icon_sets()‘ function conditionally adds an icon from the Font Awesome library (via shiny) to each cell of a column and assigns a color depending on their value in relation to other values in that particular column. Any number of icons and any number of colors can be used. The number of icons and colors determines how the values are shown from low values to high values. The icons can be positioned over, above, below, or to the right or left of the values. The size of the icon can be adjusted. Icons and icon colors can be provided via another reference column in the dataset which is useful when assigning icons/colors to particular occurrences. It should be placed within the cell argument in reactable::colDef.

## Usage

```
icon_sets(
  data,
  icons = c("circle"),
  colors = c("#67a9cf", "#808080", "#ef8a62"),
  opacity = 1,
  icon_position = "right",
  icon_ref = NULL,
  icon_size = 16,
  icon_color_ref = NULL,
  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 blue-grey-orange: c("#67a9cf","#808080","#ef8a62"). Can use R's built-in colors or other color packages. |
| `opacity` | A value between 0 and 1 that adjusts the opacity in colors. A value of 0 is fully transparent, a value of 1 is fully opaque. Default is 1. |
| `icon_position` | Position of icon relative to numbers. Options are "left", "right", above", "below", or "over". Default is right. |
| `icon_ref` | Optionally assign icons from another column by providing the name of the column containing the icons in quotes. Only one icon can be provided per cell. Default is NULL. |
| `icon_size` | A value representing the size of the icon in px. Default is 16. |
| `icon_color_ref` | Optionally assign color to the icons from another column by providing the name of the column containing the icon colors in quotes. Only one color can be provided per cell. Default is NULL. |
| `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 blue circles to the lowest-third values,
## grey circles to the middle-third values,
## and orange 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)))
```

```
## Position icons relative to the numbers. Options are to the left, right, above, below, or over.
reactable(car_prices,
defaultColDef = colDef(cell = icon_sets(car_prices,
icon_position = "above")))
```

## journal *Theme journal*

### Description

Bootstrap-inspired journal theme

### Usage

```
journal(
  font_family = "Tahoma",
  font_size = 14,
  font_color = "#222222",
  header_font_family = "Tahoma",
  header_font_size = 15,
  header_font_color = "#fad9d8",
  cell_padding = 6
)
```

### Arguments

| | |
|---|---|
| `font_family` | Font family for the text within the table. Default is Tahoma. |
| `font_size` | Numeric value representing the size of the font within the table (in px). Default is 14. |
| `font_color` | Color of the font for the text within the table and the group headers. Default is #222222. |
| `header_font_family` | Font family for the header text. Default is Tahoma. |
| `header_font_size` | Numeric value representing the size of the font within the table (in px). Default is 15. |
| `header_font_color` | Color of the font for the header text. Default is #fad9d8. |
| `cell_padding` | Numeric value representing the padding size between cells (in px). Default is 6. |

### Value

an object of class theme that is applied to a reactable table.

## Examples

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

## Standard journal theme
reactable(data,
          theme = journal())

## Cerulean theme with additional options applied
reactable(data,
          theme = journal(font_size = 12, font_color = "grey", cell_padding = 3))
```

---

`lux` *Theme lux*

---

## Description

Bootstrap-inspired lux theme

## Usage

```
lux(
  font_family = "Tahoma",
  font_size = 14,
  font_color = "#8c8c8c",
  header_font_family = "Tahoma",
  header_font_size = 15,
  header_font_color = "#7f7f7f",
  cell_padding = 6
)
```

## Arguments

| | |
|---|---|
| `font_family` | Font family for the text within the table. Default is Tahoma. |
| `font_size` | Numeric value representing the size of the font within the table (in px). Default is 14. |
| `font_color` | Color of the font for the text within the table and the group headers. Default is #8c8c8c. |
| `header_font_family` | Font family for the header text. Default is Tahoma. |
| `header_font_size` | Numeric value representing the size of the font within the table (in px). Default is 15. |
| `header_font_color` | Color of the font for the header text. Default is #7f7f7f. |
| `cell_padding` | Numeric value representing the padding size between cells (in px). Default is 6. |

### Value

an object of class theme that is applied to a reactable table.

### Examples

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

## Standard lux theme
reactable(data,
          theme = lux())

## Cerulean theme with additional options applied
reactable(data,
          theme = lux(font_size = 12, font_color = "grey", cell_padding = 3))
```

---

## `midnight` *Theme midnight*

---

### Description

midnight table theme

### Usage

```
midnight(
  font_family = "Tahoma",
  font_size = 15,
  font_color = "#727272",
  header_font_family = "Tahoma",
  header_font_size = 15,
  header_font_color = "#666666",
  cell_padding = 6
)
```

### Arguments

| | |
|---|---|
| `font_family` | Font family for the text within the table. Default is Tahoma. |
| `font_size` | Numeric value representing the size of the font within the table (in px). Default is 15. |
| `font_color` | Color of the font for the text within the table and the group headers. Default is #727272. |
| `header_font_family` | Font family for the header text. Default is Tahoma. |
| `header_font_size` | Numeric value representing the size of the font within the table (in px). Default is 15. |

| | |
|---|---|
| `header_font_color` | Color of the font for the header text. Default is #666666. |
| `cell_padding` | Numeric value representing the padding size between cells (in px). Default is 6. |

## Value

an object of class theme that is applied to a reactable table.

## Examples

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

## Standard midnight theme
reactable(data,
          theme = midnight())

## Cerulean theme with additional options applied
reactable(data,
          theme = midnight(font_size = 12, font_color = "grey", cell_padding = 3))
```

---

`midnightblue` *Theme midnightblue*

---

## Description

midnightblue table theme

## Usage

```
midnightblue(
  font_family = "Tahoma",
  font_size = 15,
  font_color = "#bababa",
  header_font_family = "Tahoma",
  header_font_size = 15,
  header_font_color = "lightgrey",
  cell_padding = 6
)
```

## Arguments

| | |
|---|---|
| `font_family` | Font family for the text within the table. Default is Tahoma. |
| `font_size` | Numeric value representing the size of the font within the table (in px). Default is 15. |
| `font_color` | Color of the font for the text within the table and the group headers. Default is #bababa. |

| | |
|---|---|
| `header_font_family` | Font family for the header text. Default is Tahoma. |
| `header_font_size` | Numeric value representing the size of the font within the table (in px). Default is 15. |
| `header_font_color` | Color of the font for the header text. Default is lightgrey. |
| `cell_padding` | Numeric value representing the padding size between cells (in px). Default is 6. |

## Value

an object of class theme that is applied to a reactable table.

## Examples

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

## Standard midnightblue theme
reactable(data,
          theme = midnightblue())

## Cerulean theme with additional options applied
reactable(data,
          theme = midnightblue(font_size = 12, font_color = "grey", cell_padding = 3))
```

---

## `minty` *Theme minty*

---

### Description

Bootstrap-inspired minty theme

### Usage

```
minty(
  font_family = "Helvetica",
  font_size = 15,
  font_color = "#9a9a9a",
  header_font_family = "Helvetica",
  header_font_size = 16,
  header_font_color = "#c9e7de",
  cell_padding = 6
)
```

## Arguments

| | |
|---|---|
| `font_family` | Font family for the text within the table. Default is Helvetica. |
| `font_size` | Numeric value representing the size of the font within the table (in px). Default is 15. |
| `font_color` | Color of the font for the text within the table and the group headers. Default is #9a9a9a. |
| `header_font_family` | Font family for the header text. Default is Helvetica. |
| `header_font_size` | Numeric value representing the size of the font within the table (in px). Default is 16. |
| `header_font_color` | Color of the font for the header text. Default is #c9e7de. |
| `cell_padding` | Numeric value representing the padding size between cells (in px). Default is 6. |

## Value

an object of class theme that is applied to a reactable table.

## Examples

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

## Standard minty theme
reactable(data,
          theme = minty())

## Cerulean theme with additional options applied
reactable(data,
          theme = minty(font_size = 12, font_color = "grey", cell_padding = 3))
```

---

`nytimes` *Theme nytimes*

---

## Description

The New York Times-inspired table theme

## Usage

```
nytimes(
  font_family = "Helvetica",
  font_size = 13,
  font_color = "#333333",
  header_font_family = "Helvetica",
  header_font_size = 11,
  header_font_color = "#999999",
  cell_padding = 5
)
```

## Arguments

| | |
|---|---|
| `font_family` | Font family for the text within the table. Default is Helvetica. |
| `font_size` | Numeric value representing the size of the font within the table (in px). Default is 13. |
| `font_color` | Color of the font for the text within the table and the group headers. Default is #333333. |
| `header_font_family` | Font family for the header text. Default is Helvetica. |
| `header_font_size` | Numeric value representing the size of the font within the table (in px). Default is 11. |
| `header_font_color` | Color of the font for the header text. Default is #999999. |
| `cell_padding` | Numeric value representing the padding size between cells (in px). Default is 5. |

## Value

an object of class theme that is applied to a reactable table.

## Examples

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

## Standard nytimes theme
reactable(data,
          theme = nytimes())

## Cerulean theme with additional options applied
reactable(data,
          theme = nytimes(font_size = 12, font_color = "grey", cell_padding = 3))
```

---

`pff` *Theme pff*

---

## Description

Pro Football Focus-inspired table theme

## Usage

```
pff(
  font_family = "Arial",
  font_size = 16,
  font_color = "#878e94",
  header_font_family = "Arial",
  header_font_size = 12,
  header_font_color = "#ffffff",
  cell_padding = 4
)
```

## Arguments

| | |
|---|---|
| `font_family` | Font family for the text within the table. Default is Arial. |
| `font_size` | Numeric value representing the size of the font within the table (in px). Default is 16. |
| `font_color` | Color of the font for the text within the table and the group headers. Default is #878e94. |
| `header_font_family` | Font family for the header text. Default is Arial. |
| `header_font_size` | Numeric value representing the size of the font within the table (in px). Default is 12. |
| `header_font_color` | Color of the font for the header text. Default is #ffffff. |
| `cell_padding` | Numeric value representing the padding size between cells (in px). Default is 4. |

## Value

an object of class theme that is applied to a reactable table.

## Examples

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

## Standard pff theme
reactable(data,
          theme = pff())

## Cerulean theme with additional options applied
reactable(data,
          theme = pff(font_size = 12, font_color = "grey", cell_padding = 3))
```

---

`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))))
```

## sandstone *Theme sandstone*

### Description

Bootstrap-inspired sandstone theme

### Usage

```
sandstone(
  font_family = "Georgia",
  font_size = 15,
  font_color = "#3e3f3a",
  header_font_family = "Georgia",
  header_font_size = 16,
  header_font_color = "#7c7a78",
  cell_padding = 6
)
```

### Arguments

| | |
|---|---|
| `font_family` | Font family for the text within the table. Default is Georgia. |
| `font_size` | Numeric value representing the size of the font within the table (in px). Default is 15. |
| `font_color` | Color of the font for the text within the table and the group headers. Default is #3e3f3a. |
| `header_font_family` | Font family for the header text. Default is Georgia. |
| `header_font_size` | Numeric value representing the size of the font within the table (in px). Default is 16. |
| `header_font_color` | Color of the font for the header text. Default is #7c7a78. |
| `cell_padding` | Numeric value representing the padding size between cells (in px). Default is 6. |

### Value

an object of class theme that is applied to a reactable table.

### Examples

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

## Standard sandstone theme
reactable(data,
          theme = sandstone())
```

```
## Cerulean theme with additional options applied
reactable(data,
          theme = sandstone(font_size = 12, font_color = "grey", cell_padding = 3))
```

---

## `save_reactable` *Save a reactable table as an image or .html file*

---

### Description

The ‘save_reactable()‘ function converts either a reactable table, .html file, or .Rmd file to an image or .html file and saves it in the user’s working directory. Table can be saved as either a .png file or .html file. Other file types are not currently supported. If the reactable table is located within an .Rmd file and has additional CSS styles provided, specify the name of the .Rmd file as the input. Alternatively, if the reactable table exists in an .html file, specify the name of the .html file as the input. ‘save_reactable()‘ depends on the ‘webshot2‘ package which can be downloaded from https://github.com/rstudio/webshot2. Additional parameters available within webshot2::webshot such as vwidth, vheight, and cliprect can be passed through ‘save_reactable()‘. The zoom value within webshot2::webshot has already been set to 2 which uses a higher pixel rate.

### Usage

```
save_reactable(input, output, ...)
```

### Arguments

| | |
|---|---|
| `input` | A reactable table, .html file, or .Rmd file |
| `output` | A .png or .html file name for the saved image |
| `...` | Optional additional parameters passed from webshot2::webshot |

### Value

a function that converts a reactable table, .html file, or .Rmd file to an .png file or .html file and saves it in the user’s working directory.

### Examples

```
## Not run:
## Save reactable table as a png file:
iris_table <- reactable(data)
save_reactable(iris_table, "iris_table.png")

## Also works with a pipe
iris_table %>%
save_reactable("iris_table.png")

## Or save as an html file:
save_reactable(iris_table, "iris_table.html")
```

```
## If the reactable table was built in R Markdown with CSS styles applied,
## specify .Rmd file as input and save_reactable will run the file
## and save the output as an image
save_reactable("iris_table.Rmd", "iris_table.png")

## Alternatively, can do the same with an .html file
save_reactable("iris_table.html", "iris_table.png")

## End(Not run)
```

## slate *Theme slate*

### Description

Bootstrap-inspired slate theme

### Usage

```
slate(
  font_family = "Arial",
  font_size = 15,
  font_color = "#aaaaaa",
  header_font_family = "Arial",
  header_font_size = 16,
  header_font_color = "#97999b",
  cell_padding = 6
)
```

### Arguments

| | |
|---|---|
| `font_family` | Font family for the text within the table. Default is Arial. |
| `font_size` | Numeric value representing the size of the font within the table (in px). Default is 15. |
| `font_color` | Color of the font for the text within the table and the group headers. Default is #aaaaaa. |
| `header_font_family` | Font family for the header text. Default is Arial. |
| `header_font_size` | Numeric value representing the size of the font within the table (in px). Default is 16. |
| `header_font_color` | Color of the font for the header text. Default is #97999b. |
| `cell_padding` | Numeric value representing the padding size between cells (in px). Default is 6. |

### Value

an object of class theme that is applied to a reactable table.

### Examples

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

## Standard slate theme
reactable(data,
          theme = slate())

## Cerulean theme with additional options applied
reactable(data,
          theme = slate(font_size = 12, font_color = "grey", cell_padding = 3))
```

---

## `spacelab` *Theme spacelab*

---

### Description

Bootstrap-inspired spacelab theme

### Usage

```
spacelab(
  font_family = "Georgia",
  font_size = 14,
  font_color = "#8e8e8e",
  header_font_family = "Georgia",
  header_font_size = 15,
  header_font_color = "#8e8e8e",
  cell_padding = 6
)
```

### Arguments

| | |
|---|---|
| `font_family` | Font family for the text within the table. Default is Georgia. |
| `font_size` | Numeric value representing the size of the font within the table (in px). Default is 14. |
| `font_color` | Color of the font for the text within the table and the group headers. Default is #8e8e8e. |
| `header_font_family` | Font family for the header text. Default is Georgia. |
| `header_font_size` | Numeric value representing the size of the font within the table (in px). Default is 15. |

| | |
|---|---|
| `header_font_color` | Color of the font for the header text. Default is #8e8e8e. |
| `cell_padding` | Numeric value representing the padding size between cells (in px). Default is 6. |

## Value

an object of class theme that is applied to a reactable table.

## Examples

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

## Standard spacelab theme
reactable(data,
          theme = spacelab())

## Cerulean theme with additional options applied
reactable(data,
          theme = spacelab(font_size = 12, font_color = "grey", cell_padding = 3))
```

---

`sunrise` *Theme sunrise*

---

## Description

sunrise table theme

## Usage

```
sunrise(
  font_family = "Tahoma",
  font_size = 15,
  font_color = "#8069ff",
  header_font_family = "Tahoma",
  header_font_size = 15,
  header_font_color = "#8069ff",
  cell_padding = 6
)
```

## Arguments

| | |
|---|---|
| `font_family` | Font family for the text within the table. Default is Tahoma. |
| `font_size` | Numeric value representing the size of the font within the table (in px). Default is 15. |
| `font_color` | Color of the font for the text within the table and the group headers. Default is #8069ff. |

| | |
|---|---|
| `header_font_family` | Font family for the header text. Default is Tahoma. |
| `header_font_size` | Numeric value representing the size of the font within the table (in px). Default is 15. |
| `header_font_color` | Color of the font for the header text. Default is #8069ff. |
| `cell_padding` | Numeric value representing the padding size between cells (in px). Default is 6. |

## Value

an object of class theme that is applied to a reactable table.

## Examples

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

## Standard sunrise theme
reactable(data,
          theme = sunrise())

## Cerulean theme with additional options applied
reactable(data,
          theme = sunrise(font_size = 12, font_color = "grey", cell_padding = 3))
```

---

`superhero` *Theme superhero*

---

## Description

Bootstrap-inspired superhero theme

## Usage

```
superhero(
  font_family = "Georgia",
  font_size = 14,
  font_color = "#ebebeb",
  header_font_family = "Georgia",
  header_font_size = 15,
  header_font_color = "#ebebeb",
  cell_padding = 6
)
```

## Arguments

| | |
|---|---|
| `font_family` | Font family for the text within the table. Default is Georgia. |
| `font_size` | Numeric value representing the size of the font within the table (in px). Default is 14. |
| `font_color` | Color of the font for the text within the table and the group headers. Default is #ebebeb. |
| `header_font_family` | Font family for the header text. Default is Georgia. |
| `header_font_size` | Numeric value representing the size of the font within the table (in px). Default is 15. |
| `header_font_color` | Color of the font for the header text. Default is #ebebeb. |
| `cell_padding` | Numeric value representing the padding size between cells (in px). Default is 6. |

## Value

an object of class theme that is applied to a reactable table.

## Examples

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

## Standard superhero theme
reactable(data,
          theme = superhero())

## Cerulean theme with additional options applied
reactable(data,
          theme = superhero(font_size = 12, font_color = "grey", cell_padding = 3))
```

---

`void` *Theme void*

---

## Description

A table style completely void of borders and headers

## Usage

```
void(
  font_family = "Verdana",
  font_size = 14,
  font_color = "#222222",
  header_font_family = "Verdana",
  header_font_size = 15,
  header_font_color = "transparent",
  cell_padding = 6
)
```

## Arguments

| | |
|---|---|
| `font_family` | Font family for the text within the table. Default is Verdana. |
| `font_size` | Numeric value representing the size of the font within the table (in px). Default is 14. |
| `font_color` | Color of the font for the text within the table. Default is #222222. |
| `header_font_family` | Font family for the header text. Default is Verdana. |
| `header_font_size` | Numeric value representing the size of the font within the table (in px). Default is 15. |
| `header_font_color` | Color of the font for the header text. Default is transparent |
| `cell_padding` | Numeric value representing the padding size between cells (in px). Default is 6. |

## Value

an object of class theme that is applied to a reactable table.

## Examples

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

## Standard void theme
reactable(data,
          theme = void())

## Cerulean theme with additional options applied
reactable(data,
          theme = void(font_size = 12, font_color = "grey", cell_padding = 3))
```

# Index