Package ‘condformat’

October 29, 2018

Type Package
Title Conditional Formatting in Data Frames
Version 0.8.0
Date 2018-10-27
URL http://github.com/zeehio/condformat
BugReports http://github.com/zeehio/condformat/issues

Description Apply and visualize conditional formatting to data frames in R.

It renders a data frame with cells formatted according to
criteria defined by rules, using a tidy evaluation syntax. The table is
printed either opening a web browser or within the 'RStudio' viewer if
available. The conditional formatting rules allow to highlight cells
matching a condition or add a gradient background to a given column. This
package supports both 'HTML' and 'LaTeX' outputs in 'knitr' reports, and
exporting to an 'xlsx' file.

License BSD_3_clause + file LICENSE
LazyData TRUE

NeedsCompilation no

Imports grDevices, graphics, gridExtra (>= 2.3), gtable (>= 0.2.0),
htmlTable (>= 1.9), htmltools (>= 0.3.6), tibble (>= 1.3.4),
scales (>= 0.5.0), dplyr (>= 0.7.4), lazyeval (>= 0.2.0), knitr
(>= 1.20), rmarkdown (>= 0.9.6), magrittr (>= 1.5), rlang (>=
0.2.2), tidyselect (>= 0.2.2)

Suggests promises, shiny (>= 1.0.5), testthat (>= 1.0), xlsx (>=
0.6.1)

VignetteBuilder knitr

Encoding UTF-8

RoxygenNote 6.1.0

Author Sergio Oller Moreno [aut, cph, cre]
<https://orcid.org/0000-0002-8994-1549>}
Maintainer Sergio Oller Moreno <sergioller@gmail.com>
Repository CRAN

Date/Publication 2018-10-29 14:10:03 UTC
R topics documented:

+.condformat_tbl .............................................. 3
cf_field_to_css ............................................. 3
cf_field_to_gtable .......................................... 4
cf_field_to_latex ........................................... 4
condformat .................................................... 5
condformat-shiny ............................................. 6
condformat2excel ............................................. 6
condformat2grob ............................................. 7
condformat2html .............................................. 8
condformat2latex ............................................ 8
condformat2widget ........................................... 9
knit_print.condformat_tbl .............................. 9
print.condformat_tbl ......................................... 10
rule_css ...................................................... 10
rule_fill_bar ............................................... 11
rule_fill_discrete ......................................... 12
rule_fill_discrete_ ....................................... 13
rule_fill_discrete_old .................................... 15
rule_fill_gradient ......................................... 16
rule_fill_gradient2 ....................................... 17
rule_fill_gradient2_ ..................................... 18
rule_fill_gradient2_old .................................. 19
rule_fill_gradient_ ....................................... 20
rule_fill_gradient_old .................................. 21
rule_text_bold .............................................. 22
rule_text_color ............................................. 23
show_columns ............................................... 24
show_columns_ .............................................. 25
show_columns_old ......................................... 25
show_rows .................................................. 26
show_rows_old ............................................... 27
theme_caption ............................................... 28
theme_grob .................................................. 29
theme_htmlTable .......................................... 29
theme_htmlWidget ......................................... 30
theme_kable ............................................... 30

Index 32
+.condformat_tbl

**Description**

This is deprecated

**Usage**

```r
## S3 method for class 'condformat_tbl'
x + obj
```

**Arguments**

- `x`: A condformat_tbl object
- `obj`: A condformat_show or a condformat_rule object to be combined. Any other type of object will be added as expected to the data frame.

**Value**

`x`, with extended condformat_tbl attributes

**Examples**

```r
data(iris)
condformat(iris[1:5,]) + show_columns(Species)
```

---

**cf_field_to_css**

*How to export a cf_field to CSS*

**Description**

This method is exported so package users can generate their own rules

**Usage**

```r
cf_field_to_css(cf_field, xview, css_fields, unlocked)
```

**Arguments**

- `cf_field`: A cf_field object. This is like a rule, but with the computed colour values. It usually maps one-to-one to a CSS field.
- `xview`: A data frame with the columns to be printed and rows filtered
- `css_fields`: A list of matrices. The names of the list are CSS attributes and each matrix is of the size of xview and contains the respective CSS values.
- `unlocked`: A logical matrix of cells unlocked (that can still be modified by further rules).
Value

A list with two elements: css_fields and unlocked (with updated values)

cf_field_to_gtable  How to export a cf_field to grob

Description

This method is exported so package users can generate their own rules

Usage

cf_field_to_gtable(cf_field, xview, gridobj, unlocked, has_rownames, has_colnames)

Arguments

cf_field  A cf_field object. This is like a rule, but with the computed colour values. It usually maps one-to-one to a CSS field.

xview  A data frame with the columns to be printed and rows filtered

gridobj  The tableGrob object

unlocked  A logical matrix of cells unlocked (that can still be modified by further rules).

has_rownames  Whether or not the gridobj has a first column with row names

has_colnames  Whether or not the gridobj has a first row with column names

Value

A list with two elements: gridobj and unlocked (with updated values)

cf_field_to_latex  How to export cf values to latex

Description

How to export cf values to latex

Usage

cf_field_to_latex(cf_field, xview, unlocked)
condformat

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cf_field</td>
<td>A cf_field object. This is like a rule, but with the computed colour values. It usually maps one-to-one to a CSS field.</td>
</tr>
<tr>
<td>xview</td>
<td>A data frame with the columns to be printed and rows filtered</td>
</tr>
<tr>
<td>unlocked</td>
<td>A logical matrix of cells unlocked (that can still be modified by further rules).</td>
</tr>
</tbody>
</table>

Value

A list with two character matrices named before and after. Both of these matrices must be of the same size as xview.

condformat | Conditional formatting for data frames

Description

A condformat_tbl object is a data frame with attributes regarding the formatting of their cells, that can be viewed when the condformat_tbl object is printed.

Usage

condformat(x)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>A matrix or data.frame</td>
</tr>
</tbody>
</table>

Value

The condformat_tbl object. This object can be piped to apply conditional formatting rules. It can also be used as a conventional data frame.

The condformat_tbl print method generates an htmlTable, to be viewed using RStudio Viewer or an HTML browser, as available.

Examples

data(iris)
condformat(iris[1:5,])

condformat(iris[1:5,]) %>% rule_fill_gradient(Sepal.Length)

condformat(iris[1:5,]) %>%
rule_fill_discrete(Sepal.Length, expression=Sepal.Width > 2)
**condformat-shiny**  
*Shiny bindings for condformat*

**Description**

Output and render functions for using condformat within Shiny applications and interactive Rmd documents.

**Usage**

```r
condformatOutput(outputId, ...)  
renderCondformat(expr, env = parent.frame(), quoted = FALSE)  
condformat_example(display.mode = "normal")
```

**Arguments**

- **outputId**: output variable to read from
- **...**: arguments passed to htmlOutput
- **expr**: An expression that generates a condformat object
- **env**: The environment in which to evaluate expr.
- **quoted**: Is expr a quoted expression (with quote())? This is useful if you want to save an expression in a variable.
- **display.mode**: The mode in which to display the application. If set to the value "showcase", shows application code and metadata from a DESCRIPTION file in the application directory alongside the application. If set to "normal", displays the application normally. Defaults to "auto", which displays the application in the mode given in its DESCRIPTION file, if any.

**condformat2excel**  
*Writes the table to an Excel workbook*

**Description**

Writes the table to an Excel workbook

**Usage**

```r
condformat2excel(x, filename, sheet_name = "Sheet1",  
overwrite_wb = FALSE, overwrite_sheet = TRUE)
```
condformat2grob

Arguments

x A condformat_tbl object
filename The xlsx file name.
sheet_name The name of the sheet where the table will be written
overwrite_wb logical to overwrite the workbook file
overwrite_sheet logical to overwrite the sheet

condformat2grob  Converts the table to a grid object

Description

Converts the table to a grid object

Usage

condformat2grob(x)

Arguments

x A condformat_tbl object

Value

the grid object

Examples

library(condformat)
data.frame(Student = c("Alice", "Bob", "Charlie"),
    Evaluation = c("Great", "Well done", "Good job!")) %>%
condformat %>%
condformat2grob
\texttt{condformat2html} \hspace{1cm} \textit{Converts the table to a htmlTable object}

\textbf{Description}

Converts the table to a htmlTable object

\textbf{Usage}

\texttt{condformat2html(x)}

\textbf{Arguments}

\begin{itemize}
  \item \texttt{x} \hspace{1cm} A \texttt{condformat\_tbl} object
\end{itemize}

\textbf{Value}

the htmlTable object

\textbf{Examples}

\begin{verbatim}
data(iris)
condformat2html(condformat(iris[1:5,]))
\end{verbatim}

\texttt{condformat2latex} \hspace{1cm} \textit{Converts the table to LaTeX code}

\textbf{Description}

Converts the table to LaTeX code

\textbf{Usage}

\texttt{condformat2latex(x)}

\textbf{Arguments}

\begin{itemize}
  \item \texttt{x} \hspace{1cm} A \texttt{condformat\_tbl} object
\end{itemize}

\textbf{Value}

A character vector of the table source code
condformat2widget

Converting the table to a htmlTableWidget

Description
Converting the table to a htmlTableWidget

Usage
condformat2widget(x, ...)

Arguments
x A condformat_tbl object
... Deprecated: Arguments passed to htmlTable::htmlTableWidget

Value
the htmlTable widget

Examples
## Not run:
data(iris)
condformat2widget(condformat(iris[1:5,]))
## End(Not run)

knit_print.condformat_tbl

Print method for knitr, exporting to HTML or LaTeX as needed

Description
Print method for knitr, exporting to HTML or LaTeX as needed

Usage
## S3 method for class 'condformat_tbl'
knit_print(x, ...)

Arguments
x Object to print
... On a LaTeX output these are unused. On an HTML output can have "paginate=TRUE" or "paginate = FALSE"
print.condformat_tbl  

Prints the data frame in an html page and shows it.

Description

Prints the data frame in an html page and shows it.

Usage

```r
## S3 method for class 'condformat_tbl'
print(x, ..., paginate = TRUE)
```

Arguments

- `x` A condformat_tbl object
- `...` Arguments passed on to `htmltools::html_print`
  - `background` Background color for web page
  - `viewer` A function to be called with the URL or path to the generated HTML page. Can be `NULL`, in which case no viewer will be invoked.
- `paginate` A logical value. If `TRUE` the printing will be paginated

Value

the value returned by `htmlTable`

Examples

```r
data(iris)
print(condformat(iris[1:5,]))
```

---

rule_css  

Apply a CSS style property as a conditional formatting rule

Description

Apply a CSS style property as a conditional formatting rule

Usage

```r
rule_css(x, columns, expression, css_field, na.value = "", lockcells = FALSE)
```
**Arguments**

- **x**
  A condformat object, typically created with `condformat()`

- **columns**
  A character vector with column names to be coloured. Optionally `tidyselect::select_helpers()` can be used.

- **expression**
  This expression should evaluate to an array of the values

- **css_field**
  CSS style property name (e.g. "color")

- **na.value**
  CSS property value to be used in missing values (e.g. "grey")

- **lockcells**
  Logical value determining if no further rules should be applied to the affected cells.

**See Also**

Other rule: `rule_fill_bar, rule_fill_discrete, rule_fill_gradient2, rule_fill_gradient, rule_text_bold, rule_text_color`

**Examples**

```r
data(iris)
condformat(iris[c(1:5, 51:55, 101:105),]) %>%
  rule_css(Species, expression = ifelse(Species == "setosa", "red", "darkgreen"),
           css_field = "color")
```

---

**rule_fill_bar**  
*Fill column with a bar of a length proportional to a value*

**Description**

Fills the background of a column cell using a bar proportional to the value of the cell

**Usage**

```r
rule_fill_bar(x, columns, expression, low = "darkgreen",
              high = "white", background = "white", na.value = "gray",
              limits = NA, lockcells = FALSE)
```

**Arguments**

- **x**
  A condformat object, typically created with `condformat()`

- **columns**
  A character vector with column names to be coloured. Optionally `tidyselect::select_helpers()` can be used.

- **expression**
  An expression to be evaluated with the data. It should evaluate to a numeric vector, that will be used to determine the colour gradient level.

- **low**
  Colour for the beginning of the bar

- **high**
  Colour for the end of the bar
rule_fill_discrete

background  Background colour for the cell
na.value    Colour for missing values
limits      range of limits that the gradient should cover
lockcells   logical value determining if no further rules should be applied to the affected cells.

Value
The condformat_tbl object, with the added formatting information

See Also
Other rule: rule_css, rule_fill_discrete, rule_fill_gradient2, rule_fill_gradient, rule_text_bold, rule_text_color

Examples
data(iris)
condformat(iris[c(1:5, 70:75, 120:125), ]) %>% rule_fill_bar("Sepal.Length")

rule_fill_discrete  Fill column with discrete colors

Description
Fills a column or columns of a data frame using a discrete colour palette, based on an expression.

Usage
rule_fill_discrete(x, columns, expression, colours = NA,
                   na.value = "#FFFFFF", h = c(0, 360) + 15, c = 100, l = 65,
                   h.start = 0, direction = 1, lockcells = FALSE,
                   ...)  

Arguments
x          A condformat object, typically created with condformat()
columns    A character vector with column names to be coloured. Optionally tidyselect::select_helpers() can be used.
expression an expression to be evaluated with the data. It should evaluate to a logical or an integer vector, that will be used to determine which cells are to be coloured.
colours    a character vector with colours as values and the expression possible results as names.
na.value   a character string with the CSS color to be used in missing values
h          range of hues to use, in [0, 360]
c          chroma (intensity of colour), maximum value varies depending on combination of hue and luminance.
rule_fill_discrete_

1  luminance (lightness), in [0, 100]

h.start  hue to start at
direction  direction to travel around the colour wheel, 1 = clockwise, -1 = counter-clockwise
lockcells  logical value determining if no further rules should be applied to the affected cells.

Details

The syntax in condformat rules has changed since v0.7. See rule_fill_discrete_old() to transition from the old syntax to the new one.

Value

The condformat_tbl object, with the added formatting information.

See Also

Other rules: rule_css, rule_fill_bar, rule_fill_gradient2, rule_fill_gradient, rule_text_bold, rule_text_color.

Examples

data(iris)
condformat(iris[1:5, 70:75, 120:125], list)
rule_fill_discrete("Species", colours = c("setosa" = "red",
   "versicolor" = "blue",
   "virginica" = "green"))
rule_fill_discrete("Sepal.Length", expression = Sepal.Length > 4.6,
colours = c("TRUE" = "red"))

rule_fill_discrete(c(starts_with("Sepal")), starts_with("Petal"),
expression = Sepal.Length > 4.6,
colours = c("TRUE" = "red"))

rule_fill_discrete_  Fill column with discrete colors (deprecated)

Description

This is a deprecated function.

Usage

rule_fill_discrete_(columns, expression = ~., colours = NA, h = c(0, 360) + 15, c = 100, l = 65, h.start = 0, direction = 1,
na.value = "#FFFFFF", lockcells = FALSE)
Arguments

- **columns**: a character vector with the column names or a list with dplyr select helpers given as formulas or a combination of both.
- **expression**: a formula to be evaluated with the data that will be used to determine which cells are to be coloured. See the examples to use it programmatically.
- **colours**: a character vector with colours as values and the expression possible results as names.
- **h**: range of hues to use, in [0, 360]
- **c**: chroma (intensity of colour), maximum value varies depending on combination of hue and luminance.
- **l**: luminance (lightness), in [0, 100]
- **h.start**: hue to start at
- **direction**: direction to travel around the colour wheel, 1 = clockwise, -1 = counter-clockwise
- **na.value**: a character string with the CSS color to be used in missing values
- **lockcells**: logical value determining if no further rules should be applied to the affected cells.

Examples

```r
data(iris)
condformat(iris[c(1,51,101), ]) +
  rule_fill_discrete_(columns=c("Species"))
condformat(iris[c(1,51,101), ]) +
  rule_fill_discrete_("Species", expression=~Sepal.Length > 6)

# Use it programmatically:
color_column_larger_than_threshold <- function(x, column, threshold) {
  condformat(x) +
  rule_fill_discrete_(column,
    expression=as.name(column) > threshold)
}
color_column_larger_than_threshold(iris[c(1,51,101), ], "Sepal.Length", 6.3)

condformat(iris[c(1,51,101), ]) +
  rule_fill_discrete_(columns = list(~dplyr::starts_with("Petal"), "Species"),
    expression=~Species)

# Custom discrete color values can be specified with a function. The function takes
# the whole column and returns a vector with the colours.
color_pick <- function(column) {
  sapply(column,
    FUN = function(value) {
      if (value < 4.7) {
        return("red")
      } else if (value < 5.0) {
        return("yellow")
      } else {
        return("green")
      }
    })
}
rule_fill_discrete_old

```r
condformat(head(iris)) +
  rule_fill_discrete_("Sepal.Length", ~ color_pick(Sepal.Length), colours = identity)
```

---

rule_fill_discrete_old

_Fill column with discrete colors (deprecated)_

### Description

Fills a column or columns of a data frame using a discrete colour palette, based on an expression.

### Usage

```r
rule_fill_discrete_old(..., expression, colours = NA,
  na.value = "#FFFFFF", h = c(0, 360) + 15, c = 100, l = 65,
  h.start = 0, direction = 1, lockcells = FALSE)
```

### Arguments

- `...` Comma separated list of unquoted column names. If `expression` is also given, then this list can use any of the `select` syntax possibilities.
- `expression` an expression to be evaluated with the data. It should evaluate to a logical or an integer vector, that will be used to determine which cells are to be coloured.
- `colours` a character vector with colours as values and the expression possible results as names.
- `na.value` a character string with the CSS color to be used in missing values
- `h` range of hues to use, in [0, 360]
- `c` chroma (intensity of colour), maximum value varies depending on combination of hue and luminance.
- `l` luminance (lightness), in [0, 100]
- `h.start` hue to start at
- `direction` direction to travel around the colour wheel, 1 = clockwise, -1 = counter-clockwise
- `lockcells` logical value determining if no further rules should be applied to the affected cells.

### Value

The `condformat_tbl` object, with the added formatting information
Examples

data(iris)
  condformat(iris[c(1:5, 70:75, 120:125), ]) +
  rule_fill_discrete(Species, colours = c("setosa" = "red",
          "versicolor" = "blue",
          "virginica" = "green")) +
  rule_fill_discrete(Sepal.Length, expression = Sepal.Length > 4.6,
          colours = c("TRUE" = "red"))

rule_fill_gradient

Fill column with sequential colour gradient

Description

Fills the background color of a column using a gradient based on the values given by an expression

Usage

rule_fill_gradient(x, columns, expression, low = "#132B43",
                  high = "#56B1F7", space = "Lab", na.value = "#7F7F7F",
                  limits = NA, lockcells = FALSE, 
)

Arguments

  x         A condformat object, typically created with `condformat()`
  columns   A character vector with column names to be coloured. Optionally `tidyselect::select_helpers()` can be used.
  expression an expression to be evaluated with the data. It should evaluate to a numeric vector, that will be used to determine the colour gradient level.
  low       colour for low end of gradient.
  high      colour for high end of gradient.
  space     colour space in which to calculate gradient. Must be "Lab" - other values are deprecated.
  na.value  fill color for missing values
  limits    range of limits that the gradient should cover
  lockcells logical value determining if no further rules should be applied to the affected cells.
  ...       Dots are used to transition from the old syntax `rule_fill_discrete_old` to the new one

Details

The syntax in condformat rules has changed since v0.7. See `rule_fill_gradient_old`
*rule_fill_gradient2*

**Value**

The condformat_tbl object, with the added formatting information

**See Also**

Other rule: `rule_css, rule_fill_bar, rule_fill_discrete, rule_fill_gradient2, rule_text_bold, rule_text_color`

**Examples**

```r
data(iris)
condformat(iris[c(1:5, 70:75, 120:125), ]) %>%
  rule_fill_gradient(Sepal.Length) %>%
  rule_fill_gradient(Species, expression = Sepal.Length - Sepal.Width)

condformat(iris[c(1:5, 70:75, 120:125), ]) %>%
  rule_fill_gradient("Petal.Length") %>%
  rule_fill_gradient(starts_with("Sepal"), expression = Sepal.Length - Sepal.Width)
```

---

**rule_fill_gradient2**  
*Fill column with sequential color gradient*

**Description**

Fills the background color of a column using a gradient based on the values given by an expression

**Usage**

```r
rule_fill_gradient2(x, columns, expression, low = scales::muted("red"),
                    mid = "white", high = scales::muted("blue"), midpoint = NA,
                    space = "Lab", na.value = "#7F7F7F", limits = NA,
                    lockcells = FALSE, ...)
```

**Arguments**

- `x`  
  A condformat object, typically created with `condformat()`

- `columns`  
  A character vector with column names to be colored. Optionally `tidyselect::select_helpers()` can be used.

- `expression`  
  An expression to be evaluated with the data. It should evaluate to a logical or an integer vector, that will be used to determine which cells are to be colored.

- `low`  
  Colour for lower end of gradient.

- `mid`  
  Colour for middle point.

- `high`  
  Colour for higher end of gradient.

- `midpoint`  
  The value used for the middle color (the median by default)
space colour space in which to calculate gradient. Must be "Lab" - other values are deprecated.
na.value fill color for missing values
limits range of limits that the gradient should cover
lockcells logical value determining if no further rules should be applied to the affected cells.
... Dots are used to transition from the old syntax `rule_fill_discrete_old()` to the new one

Details
The syntax in condformat rules has changed since v0.7. See `rule_fill_gradient_old()`

Value
The `condformat_tbl` object, with the added formatting information

See Also
Other rule: `rule_css, rule_fill_bar, rule_fill_discrete, rule_fill_gradient, rule_text_bold, rule_text_color`

Examples
```r
data(iris)
condformat(iris[c(1:5, 70:75, 120:125), ]) %>%
  rule_fill_gradient2(Sepal.Length) %>%
  rule_fill_gradient2(Species, expression=Sepal.Length - Sepal.Width)

condformat(iris[c(1:5, 70:75, 120:125), ]) %>%
  rule_fill_gradient2("Petal.Length") %>%
  rule_fill_gradient2(starts_with("Sepal"), expression=Sepal.Length - Sepal.Width)
```

---

**rule_fill_gradient2**  
*Fill column with divergent color gradient (deprecated)*

Description
Fills the background color of a column using a three colors gradient based on the values of an expression

Usage
```r
rule_fill_gradient2_(columns, expression = ~.,
  low = scales::muted("red"), mid = "white",
  high = scales::muted("blue"), midpoint = NA, space = "Lab",
  na.value = "#FF7F7F", limits = NA, lockcells = FALSE)
```
Arguments

- **columns**: a character vector with the column names or a list with dplyr select helpers given as formulas or a combination of both.
- **expression**: a formula to be evaluated with the data that will be used to determine which cells are to be colored. See the examples to use it programmatically.
- **low**: colour for low end of gradient.
- **mid**: colour for mid point.
- **high**: colour for high end of gradient.
- **midpoint**: the value used for the middle color (the median by default).
- **space**: colour space in which to calculate gradient. Must be "Lab" - other values are deprecated.
- **na.value**: fill color for missing values.
- **limits**: range of limits that the gradient should cover.
- **lockcells**: logical value determining if no further rules should be applied to the affected cells.

Examples

```r
data(iris)
condformat(iris[1:10,1] + rule_fill_gradient2_.(columns=c("Sepal.Length")))
condformat(iris[1:10,1] + rule_fill_gradient2_.("Species",
  expression= -Sepal.Length-Sepal.Width))

# Use it programmatically
color_column <- function(x, column) {
  condformat(x) +
  rule_fill_gradient2_.(column, expression= ~ uq(as.name(column)))
}
color_column(iris[c(1,5,10),], "Sepal.Length")
```

---

**rule_fill_gradient2_old**

*Fill column with divergent color gradient (deprecated)*

**Description**

Fills the background color of a column using a three colors gradient based on the values of an expression.

**Usage**

```r
rule_fill_gradient2_old(..., expression, low = scales::muted("red"),
  mid = "white", high = scales::muted("blue"), midpoint = NA,
  space = "Lab", na.value = "#7F7F7F", limits = NA,
  lockcells = FALSE)
```
Arguments

... Comma separated list of unquoted column names. If expression is also given, then this list can use any of the select syntax possibilities.

expression an expression to be evaluated with the data. It should evaluate to a numeric vector, that will be used to determine the color gradient level.

low colour for low end of gradient.

mid colour for mid point

high colour for high end of gradient.

midpoint the value used for the middle color (the median by default)

space colour space in which to calculate gradient. Must be "Lab" - other values are deprecated.

na.value fill color for missing values

limits range of limits that the gradient should cover

lockcells logical value determining if no further rules should be applied to the affected cells.

Value

The condformat_tbl object, with the added formatting information

Examples

data(iris)
condformat(iris[[c(1:5, 70:75, 120:125), ] ] +
rule_fill_gradient2(Sepal.Length) +
rule_fill_gradient2(Species, expression=Sepal.Length - Sepal.Width)

rule_fill_gradient_ Fill column with sequential colour gradient (deprecated)

Description

Fills the background color of a column using a gradient based on the values given by an expression

Usage

rule_fill_gradient_(columns, expression = ~., low = "#132B43", high = "#56B1F7", space = "Lab", na.value = "#F7F7F7", limits = NA, lockcells = FALSE)
rule_fill_gradient_old

Fill column with sequential colour gradient (deprecated)

Description

Fills the background color of a column using a gradient based on the values given by an expression

Usage

rule_fill_gradient_old(..., expression, low = "#132B43", high = "#56B1F7", space = "Lab", na.value = "#7F7F7F", limits = NA, lockcells = FALSE)
Arguments

... Comma separated list of unquoted column names. If expression is also given, then this list can use any of the select syntax possibilities.

eexpression an expression to be evaluated with the data. It should evaluate to a numeric vector, that will be used to determine the colour gradient level.

dlow colour for low end of gradient.

dhigh colour for high end of gradient.

dspace colour space in which to calculate gradient. Must be "Lab" - other values are deprecated.

dna.value fill color for missing values

dlimits range of limits that the gradient should cover

dlockcells logical value determining if no further rules should be applied to the affected cells.

Value

The condformat_tbl object, with the added formatting information

Examples

data(iris)
condformat(iris[c(1:5, 70:75, 120:125), ]) +
  rule_fill_gradient(Sepal.Length) +
  rule_fill_gradient(Species, expression=Sepal.Length - Sepal.Width)

rule_text_bold Use bold text if a condition is met

Description

Use bold text if a condition is met

Usage

rule_text_bold(x, columns, expression, na.bold = FALSE, 
lockcells = FALSE)

Arguments

x A condformat object, typically created with condformat()

columns A character vector with column names to be coloured. Optionally tidyselect::select_helpers() can be used.

eexpression Condition that evaluates to TRUE for the rows where bold text should be applied.

dna.bold If TRUE, make missing values bold.

dlockcells logical value determining if no further rules should be applied to the affected cells.
See Also

Other rule: `rule_css`, `rule_fill_bar`, `rule_fill_discrete`, `rule_fill_gradient`, `rule_fill_gradient2`, `rule_text_color`

Examples

```r
data(iris)
condformat(iris[c(1:5, 51:55, 101:105),]) %>%
  rule_text_color(Species, expression = Species == "setosa")
```

---

**rule_text_color**

*Give a color to the text according to some expression*

**Description**

Give a color to the text according to some expression

**Usage**

```r
rule_text_color(x, columns, expression, na.color = "", lockcells = FALSE)
```

**Arguments**

- **x**: A condformat object, typically created with `condformat()`
- **columns**: A character vector with column names to be coloured. Optionally `tidyselect::select_helpers()` can be used.
- **expression**: Condition that evaluates to color names for the rows where text should be coloured
- **na.color**: Color for missing values
- **lockcells**: logical value determining if no further rules should be applied to the affected cells.

**See Also**

Other rule: `rule_css`, `rule_fill_bar`, `rule_fill_discrete`, `rule_fill_gradient`, `rule_fill_gradient2`, `rule_text_bold`

**Examples**

```r
data(iris)
condformat(iris[c(1:5, 51:55, 101:105),]) %>%
  rule_text_color(Species, expression = ifelse(Species == "setosa", "blue", ""))
```
show_columns  Selects the variables to be printed

Description

Keeps the variables you mention in the printed table. Compared to `select`, `show_columns` does not remove the columns from the data frame, so formatting rules can still depend on them.

Usage

```r
show_columns(x, columns, col_names, ...)
```

Arguments

- `x`: A condformat object, typically created with `condformat()`
- `columns`: A character vector with column names to be to show. It can also be an expression can be used that will be parsed like in `tidyselect::vars_select()`. See examples.
- `col_names`: Character vector with the column names for the selected columns
- `...`: Dots are used to transition from the old syntax `show_columns_old()` to the new one

Value

The condformat object with the rule added

See Also

- `select`

Examples

```r
data(iris)
x <- head(iris)

# Include some columns:
condformat(x) %>% show_columns(c(Sepal.Length, Sepal.Width, Species))
condformat(x) %>% show_columns(c("Sepal.Length", "Sepal.Width", "Species"))

# Rename columns:
condformat(x) %>
  show_columns(c(Sepal.Length, Species),
              col_names = c("Length", "Spec."))

# Exclude some columns:
condformat(x) %>% show_columns(c(-Petal.Length, -Petal.Width))
```
show_columns_  

condformat(x) %>% show_columns(c(starts_with("Petal"), Species))

petal_width <- "Petal.Width"
condformat(x) %>% show_columns(! petal_width)

show_columns_  

Description
Show columns (deprecated)

Usage
show_columns_(..., .dots, col_names)

Arguments
...  Comma separated list of unquoted expressions
.dots A character vector with columns to show
col_names Character vector with the column names for the selected columns

Examples

data(iris)
x <- head(iris)
# Use standard evaluation (columns as strings):
condformat(x) +
  show_columns_(.dots = c("Sepal.Length", "Species"), col_names = c("Sepal Length", "Species"))

show_columns_old  

Description
This syntax is deprecated and show_columns should be used instead

Usage
show_columns_old(..., col_names)

Arguments
...  Comma separated list of unquoted expressions
col_names Character vector with the column names for the selected columns
show_rows

Details

Keeps the variables you mention in the printed table. Compared to `select`, `show_columns` does not remove the columns from the data frame, so formatting rules can still depend on them.

Value

A `condformat_show_columns` object, usually to be added to a `condformat_tbl` object

See Also

`select`

Examples

```r
library(dplyr) # for starts_with()
data(iris)
x <- head(iris)

# Include some columns:
condformat(x) + show_columns(Sepal.Length, Sepal.Width, Species)

# Rename columns:
condformat(x) + show_columns(Sepal.Length, Species, col_names = c("Length", "Spec."))

# Exclude some columns:
condformat(x) + show_columns(-Petal.Length, -Petal.Width)

# Select columns using dplyr syntax:
condformat(x) + show_columns(starts_with("Petal"), Species)
```

show_rows

*Selects the rows to be printed*

Description

Keeps the rows you mention in the printed table. Compared to `filter`, `show_rows` does not remove the rows from the actual data frame, they are removed only for printing.

Usage

`show_rows(x, ...)`

Arguments

x  

condformat_tbl object

...  

Expressions used for filtering
### show_rows_old

**Value**

A `condformat_show_rows` object, usually to be added to a `condformat_tbl` object as shown in the examples.

**See Also**

`filter`

**Examples**

```r
library(condformat)
data(iris)
x <- head(iris)
condformat(x) %>% show_rows(Sepal.Length > 4.5, Species == "setosa")
# Use it programatically
expr_as_text <- 'Sepal.Length > 4.5'
expr <- rlang::parse_expr(expr_as_text)
condformat(x) %>% show_rows(! expr)
# With multiple arguments:
expr_as_text <- c('Sepal.Length > 4.5', 'Species == "setosa"')
exprs <- lapply(expr_as_text, rlang::parse_expr)
condformat(x) %>% show_rows(! exprs)
```

---

**show_rows_old**  
*Selects the rows to be printed (deprecated)*

**Description**

This function is deprecated. Use `show_rows` instead.

**Usage**

```
show_rows_old(...)  
show_rows_(..., .dots)
```

**Arguments**

- `...`  
  Expressions used for filtering
- `.dots`  
  A list of lazy objects. See examples

**Details**

Keeps the rows you mention in the printed table. Compared to `filter`, `show_rows` does not remove the rows from the actual data frame, they are removed only for printing.
theme_caption

Sets the caption of a condformat object

Description

The advantage with respect to theme_htmlTable(caption = "My table") is that this works with HTML and LaTeX outputs

Usage

theme_caption(x, caption = "")

Arguments

x The condformat object
caption The caption to show

Examples

data(iris)
condformat(head(iris)) %>%
  theme_caption(caption = "My Caption")
theme_grob

Customizes appearance of condformat object

Description
This is only used on grob output.

Usage
theme_grob(x, ...)

Arguments
x The condformat object
... Arguments to be passed to gridExtra::tableGrob (see examples)

See Also
tableGrob

Examples
data(iris)
condformat(head(iris)) %>%
  theme_grob(base_size = 10, base_colour = "red")

theme_htmlTable

Customizes appearance of condformat object

Description
Customizes appearance of condformat object

Usage
theme_htmlTable(x, ...)

Arguments
x The condformat object
... Arguments to be passed to htmlTable

See Also
htmlTable
Examples

```r
data(iris)
condformat(head(iris)) \%\% theme_htmlTable(caption="Table 1: My iris table", rnames=FALSE)
```

theme_htmlWidget  

*Customizes appearance of condformat object*

Description

Customizes appearance of condformat object

Usage

```r
theme_htmlWidget(x, ...)
```

Arguments

- `x`  
The condformat object
- `...`  
Arguments to be passed to htmlTable::htmlTableWidget (see examples)

See Also

htmlTable

Examples

```r
data(iris)
condformat(head(iris)) \%\% theme_htmlWidget(number_of_entries = c(10, 25, 100),
                      width = NULL, height = NULL, elementId = NULL)
```

theme_kable  

*Customizes appearance of condformat object*

Description

This is only used on LaTeX output.

Usage

```r
theme_kable(x, ...)
```

Arguments

- `x`  
The condformat object
- `...`  
Arguments to be passed to knitr::kable (see examples)
theme_kable

See Also

kable

Examples

data(iris)
condformat(head(iris)) %>%
  theme_kable(booktabs = TRUE, caption = "My Caption")
Index

+.condformat_tbl, 3
 cf_field_to_css, 3
cf_field_to_gtable, 4
cf_field_to_latex, 4
condformat, 5
condformat(), 11, 12, 16, 17, 22–24
condformat-shiny, 6
condformat2excel, 6
condformat2grob, 7
condformat2html, 8
condformat2latex, 8
condformat2widget, 9
condformat_example (condformat-shiny), 6
condformatOutput (condformat-shiny), 6

filter, 26–28

htmlTable, 29, 30

kable, 31
knit_print.condformat_tbl, 9

print.condformat_tbl, 10

renderCondformat (condformat-shiny), 6
rule_css, 10, 12, 13, 17, 18, 23
rule_fill_bar, 11, 11, 13, 17, 18, 23
rule_fill_discrete, 11, 12, 12, 17, 18, 23
rule_fill_discrete_, 13
rule_fill_discrete_old, 15, 16
rule_fill_discrete_old(), 13, 18
rule_fill_gradient, 11–13, 16, 18, 23
rule_fill_gradient2, 11–13, 17, 17, 23
rule_fill_gradient2_, 18
rule_fill_gradient2_old, 19
rule_fill_gradient_, 20
rule_fill_gradient_old, 16, 21
rule_fill_gradient_old(), 18
rule_text_bold, 11–13, 17, 18, 22, 23
rule_text_color, 11–13, 17, 18, 23, 23

select, 15, 20, 22, 24, 26
show_columns, 24, 25
show_columns_, 25
show_columns_old, 25
show_columns_old(), 24
show_rows, 26, 27
show_rows_(show_rows_old), 27
show_rows_old, 27

tableGrob, 29
theme_caption, 28
theme_grob, 29
theme_htmlTable, 29
theme_htmlWidget, 30
theme_kable, 30
tidyselect::select_helpers(), 11, 12, 16, 17, 22, 23
tidyselect::vars_select(), 24