

Package ‘condformat’

May 18, 2017

Type Package

Title Conditional Formatting in Data Frames

Version 0.6.0

Date 2017-05-18

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 syntax similar to 'ggplot2'. 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 htmlTable (>= 1.9), htmltools, tibble, scales, dplyr, lazyeval (>= 0.2.0), knitr, rmarkdown (>= 0.9.6), gplots, utils

Suggests shiny, testthat (>= 1.0), xlsx (>= 0.5.7), rJava

VignetteBuilder knitr

RoxygenNote 6.0.1

Author Sergio Oller Moreno [aut, cph, cre]

Maintainer Sergio Oller Moreno <sergioller@gmail.com>

Repository CRAN

Date/Publication 2017-05-18 15:54:52 UTC

R topics documented:

| | |
|-----------------------------|----|
| +.condformat_tbl | 2 |
| condformat | 3 |
| condformat-shiny | 4 |
| condformat2excel | 4 |
| condformat2html | 5 |
| condformat2latex | 5 |
| condformat2widget | 6 |
| get_css_field | 6 |
| knit_print.condformat_tbl | 7 |
| lock_cells | 7 |
| print.condformat_tbl | 8 |
| render_rules_condformat_tbl | 8 |
| rule_fill_discrete | 9 |
| rule_fill_discrete_ | 10 |
| rule_fill_gradient | 11 |
| rule_fill_gradient2 | 12 |
| rule_fill_gradient2_ | 13 |
| rule_fill_gradient_ | 14 |
| show_columns | 16 |
| show_rows | 17 |
| theme_htmlTable | 18 |

Index 19

| | |
|------------------|---|
| +.condformat_tbl | <i>Combines the data with the formatting rules and graphical options.</i> |
|------------------|---|

Description

Combines the data with the formatting rules and graphical options.

Usage

```
## 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

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

condformat

Converts a data frame or matrix to a condformat_tbl class.

Description

condformat_tbl objects allow to add conditional formatting information, that can be viewed when the condformat_tbl object is printed.

Usage

```
condformat(x)
```

Arguments

x A matrix or data.frame

Details

They also have all properties from conventional data frames.

Value

The condformat_tbl object. This object can be used in a ggplot-like syntax 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

```
condformatOutput(outputId, ...)
```

```
renderCondformat(expr, env = parent.frame(), quoted = FALSE)
```

```
condformat_example()
```

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. |

condformat2excel *Writes the table to an Excel sheet*

Description

Writes the table to an Excel sheet

Usage

```
condformat2excel(x, filename)
```

Arguments

| | |
|----------|-------------------------|
| x | A condformat_tbl object |
| filename | The xlsx file name |

| | |
|-----------------|---|
| condformat2html | <i>Converts the table to a htmlTable object</i> |
|-----------------|---|

Description

Converts the table to a htmlTable object

Usage

```
condformat2html(x)
```

Arguments

x A condformat_tbl object

Value

the htmlTable object

Examples

```
data(iris)
condformat2html(condformat(iris[1:5,]))
```

| | |
|------------------|---|
| condformat2latex | <i>Converts the table to LaTeX code</i> |
|------------------|---|

Description

Converts the table to LaTeX code

Usage

```
condformat2latex(x, ...)
```

Arguments

x A condformat_tbl object
... arguments passed to knitr::kable

Value

A character vector of the table source code

| | |
|-------------------|--|
| condformat2widget | <i>Converts the table to a htmlTableWidget</i> |
|-------------------|--|

Description

Converts the table to a htmlTableWidget

Usage

```
condformat2widget(x, ...)
```

Arguments

| | |
|-----|--|
| x | A condformat_tbl object |
| ... | Arguments passed to htmlTable::htmlTableWidget |

Value

the htmlTable widget

Examples

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

## End(Not run)
```

| | |
|---------------|---|
| get_css_field | <i>Returns a matrix with the value of the CSS field for each element of our data frame.</i> |
|---------------|---|

Description

Returns a matrix with the value of the CSS field for each element of our data frame.

Usage

```
get_css_field(finalformat, field)
```

Arguments

| | |
|-------------|--|
| finalformat | list with the css_fields |
| field | the name of the CSS field to be returned |

Value

a matrix of size xview with the CSS field information

```
knitr_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'
knitr_print(x, ...)
```

Arguments

| | |
|-----|--|
| x | Object to print |
| ... | Provided for knitr_print compatibility |

| | |
|------------|---|
| lock_cells | <i>If rule_locks is TRUE, locks the cells specified by mask so no further rules modify them</i> |
|------------|---|

Description

If rule_locks is TRUE, locks the cells specified by mask so no further rules modify them

Usage

```
lock_cells(rule_locks, mask, finalformat)
```

Arguments

| | |
|-------------|--|
| rule_locks | a logical. If FALSE, lock_cells does nothing |
| mask | a logical matrix. TRUE if the cell is affected by the rule |
| finalformat | The finalformat list, with the format options |

Value

finalformat, the list with format options

```
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

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

Arguments

| | |
|----------|---|
| x | A condformat_tbl object |
| ... | optional arguments to print |
| paginate | A logical value. If TRUE the printing will be paginated |

Value

the value returned by htmlTable

Examples

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

```
render_rules_condformat_tbl
```

Renders the css matrix to format the xview table

Description

Renders the css matrix to format the xview table

Usage

```
render_rules_condformat_tbl(rules, xfiltered, xview, format)
```

Arguments

| | |
|-----------|---|
| rules | List of rules to be applied |
| xfiltered | Like xview, but with all the columns (rules will use columns that won't be printed) |
| xview | Data frame with the rows and columns that will be printed |
| format | Output format (either "html" or "latex") |

Value

List with the CSS information

| | |
|--------------------|---|
| rule_fill_discrete | <i>Fill column with discrete colors</i> |
|--------------------|---|

Description

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

Usage

```
rule_fill_discrete(..., 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

See Also

Other rule: [rule_fill_discrete_](#), [rule_fill_gradient2_](#), [rule_fill_gradient2](#), [rule_fill_gradient_](#), [rule_fill_gradient](#)

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_discrete_ *Fill column with discrete colors (standard evaluation)*

Description

Fill column with discrete colors (standard evaluation)

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. |

See Also

Other rule: [rule_fill_discrete](#), [rule_fill_gradient2_](#), [rule_fill_gradient2](#), [rule_fill_gradient_](#), [rule_fill_gradient](#)

Examples

```

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=~ uq(as.name(column))> uq(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")
      }
    }
  )
}
condformat(head(iris)) +
  rule_fill_discrete_("Sepal.Length", ~ color_pick(Sepal.Length), colours = identity)

```

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(..., 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. |
| 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. |

Value

The `condformat_tbl` object, with the added formatting information

See Also

Other rule: [rule_fill_discrete_](#), [rule_fill_discrete](#), [rule_fill_gradient2_](#), [rule_fill_gradient2](#), [rule_fill_gradient_](#)

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_fill_gradient2` *Fill column with divergent colour gradient*

Description

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

Usage

```
rule_fill_gradient2(..., 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 colour 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

See Also

Other rule: [rule_fill_discrete_](#), [rule_fill_discrete](#), [rule_fill_gradient2_](#), [rule_fill_gradient_](#), [rule_fill_gradient](#)

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_gradient2_` *Fill column with divergent colour gradient (standard evaluation)*

Description

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

Usage

```
rule_fill_gradient2_(columns, expression = ~., low = scales::muted("red"),
  mid = "white", high = scales::muted("blue"), midpoint = NA,
  space = "Lab", na.value = "#7F7F7F", 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 coloured. 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. |

See Also

Other rule: [rule_fill_discrete_](#), [rule_fill_discrete](#), [rule_fill_gradient2](#), [rule_fill_gradient_](#), [rule_fill_gradient](#)

Examples

```
data(iris)
condformat(iris[1:10,]) + rule_fill_gradient2_(columns=c("Sepal.Length"))
condformat(iris[1:10,]) + 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(1,51,101),], "Sepal.Length")
```

rule_fill_gradient_ *Fill column with sequential colour gradient (standard evaluation)*

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 = "#7F7F7F", 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 coloured. See the examples to use it programmatically |
| 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. |

See Also

Other rule: [rule_fill_discrete_](#), [rule_fill_discrete](#), [rule_fill_gradient2_](#), [rule_fill_gradient2](#), [rule_fill_gradient](#)

Examples

```
data(iris)
condformat(iris[1:5,]) + rule_fill_gradient_(columns=c("Sepal.Length"))
ex1 <- condformat(iris[1:5,]) +
  rule_fill_gradient_("Species", expression=~Sepal.Length-Sepal.Width)
# Use it programmatically:
gradient_color_column1_minus_column2 <- function(x, column_to_paint, column1, column2) {
  condformat(x) +
    rule_fill_discrete_(column_to_paint,
      expression=~ uq(as.name(column1)) - uq(as.name(column2)))
}
ex2 <- gradient_color_column1_minus_column2(iris[1:5,], "Species", "Sepal.Length", "Sepal.Width")
stopifnot(ex1 == ex2)
```

| | |
|--------------|--|
| show_columns | <i>Selects the variables to be printed</i> |
|--------------|--|

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

```
show_columns(..., col_names)

show_columns_(..., .dots, col_names)
```

Arguments

| | |
|------------------------|--|
| <code>...</code> | Comma separated list of unquoted extensions. |
| <code>col_names</code> | Character vector with the column names for the selected columns |
| <code>.dots</code> | Use <code>select_()</code> to do standard evaluation. See <code>vignette("nse")</code> for details |

Value

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

See Also

[select](#)

Examples

```
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)
```



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

| | |
|-----------|---------------------------------------|
| show_rows | <i>Selects the rows to be printed</i> |
|-----------|---------------------------------------|

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(...)

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

Arguments

| | |
|--------------------|--|
| <code>...</code> | Comma separated list of unquoted extensions. |
| <code>.dots</code> | Used to work around non-standard evaluation. See <code>vignette("nse")</code> for details. |

Value

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

See Also

[filter](#)

Examples

```
library(condformat)
data(iris)
x <- head(iris)
condformat(x) + show_rows(Sepal.Length > 4.5, Species == "setosa")
library(condformat)
data(iris)
x <- head(iris)
condformat(x) + show_rows_(.dots = c("Sepal.Length > 4.5", "Species == 'setosa'"))
```

| | |
|-----------------|---|
| theme_htmlTable | <i>Customizes appearance of condformat object</i> |
|-----------------|---|

Description

Customizes appearance of condformat object

Usage

```
theme_htmlTable(...)
```

Arguments

... Arguments to be passed to htmlTable

See Also

[htmlTable](#)

Examples

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

Index

`+.condformat_tbl`, 2

`condformat`, 3

`condformat-shiny`, 4

`condformat2excel`, 4

`condformat2html`, 5

`condformat2latex`, 5

`condformat2widget`, 6

`condformat_example (condformat-shiny)`, 4

`condformatOutput (condformat-shiny)`, 4

`filter`, 17

`get_css_field`, 6

`htmlTable`, 18

`knit_print.condformat_tbl`, 7

`lock_cells`, 7

`print.condformat_tbl`, 8

`render_rules_condformat_tbl`, 8

`renderCondformat (condformat-shiny)`, 4

`rule_fill_discrete`, 9, 10, 12–15

`rule_fill_discrete_`, 9, 10, 12–15

`rule_fill_gradient`, 9, 10, 11, 13–15

`rule_fill_gradient2`, 9, 10, 12, 12, 14, 15

`rule_fill_gradient2_`, 9, 10, 12, 13, 13, 15

`rule_fill_gradient_`, 9, 10, 12–14, 14

`select`, 9, 12, 13, 16

`show_columns`, 16

`show_columns_ (show_columns)`, 16

`show_rows`, 17

`show_rows_ (show_rows)`, 17

`theme_htmlTable`, 18