

# Package ‘condformat’

July 22, 2025

**Type** Package

**Title** Conditional Formatting in Data Frames

**Version** 0.10.1

**Date** 2023-10-08

**URL** <https://zeehio.github.io/condformat/>,  
<https://github.com/zeehio/condformat>

**BugReports** <https://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

**NeedsCompilation** no

**Imports** dplyr (>= 0.7.7), grDevices, gridExtra (>= 2.3), gtable (>= 0.2.0), htmlTable (>= 1.9), htmltools (>= 0.3.6), knitr (>= 1.20), magrittr (>= 1.5), openxlsx (>= 4.1.5), rmarkdown (>= 1.10), rlang (>= 0.3.0), scales (>= 1.0.0), tibble (>= 1.3.4), tidyselect (>= 1.0.0)

**Suggests** promises, shiny (>= 1.0.5), testthat (>= 1.0), vdiff (>= 1.0.4)

**VignetteBuilder** knitr

**Encoding** UTF-8

**RoxygenNote** 7.2.3

**Author** Sergio Oller Moreno [aut, cph, cre] (ORCID:  
<<https://orcid.org/0000-0002-8994-1549>>)

**Maintainer** Sergio Oller Moreno <sergioller@gmail.com>

**Repository** CRAN

**Date/Publication** 2023-10-08 07:00:02 UTC

Contents

|                                     |           |
|-------------------------------------|-----------|
| cf_field_to_css . . . . .           | 2         |
| cf_field_to_gtable . . . . .        | 3         |
| cf_field_to_latex . . . . .         | 4         |
| condformat . . . . .                | 4         |
| condformat-shiny . . . . .          | 5         |
| condformat2excel . . . . .          | 6         |
| condformat2grob . . . . .           | 6         |
| condformat2html . . . . .           | 7         |
| condformat2latex . . . . .          | 8         |
| condformat2widget . . . . .         | 8         |
| knit_print.condformat_tbl . . . . . | 9         |
| print.condformat_tbl . . . . .      | 9         |
| rule_css . . . . .                  | 10        |
| rule_fill_bar . . . . .             | 11        |
| rule_fill_discrete . . . . .        | 12        |
| rule_fill_gradient . . . . .        | 13        |
| rule_fill_gradient2 . . . . .       | 15        |
| rule_text_bold . . . . .            | 16        |
| rule_text_color . . . . .           | 17        |
| show_columns . . . . .              | 18        |
| show_rows . . . . .                 | 19        |
| theme_caption . . . . .             | 21        |
| theme_grob . . . . .                | 21        |
| theme_htmlTable . . . . .           | 22        |
| theme_htmlWidget . . . . .          | 23        |
| theme_kable . . . . .               | 23        |
| <b>Index</b>                        | <b>25</b> |

---

|                 |                                        |
|-----------------|----------------------------------------|
| cf_field_to_css | <i>How to export a cf_field to CSS</i> |
|-----------------|----------------------------------------|

---

Description

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

Usage

```
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 | <i>How to export a cf_field to grob</i> |
|--------------------|-----------------------------------------|

---

**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 | <i>How to export cf values to latex</i> |
|-------------------|-----------------------------------------|

---

**Description**

How to export cf values to latex

**Usage**

```
cf_field_to_latex(cf_field, xview, 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                                                           |
| unlocked | A logical matrix of cells unlocked (that can still be modified by further rules).                                       |

**Value**

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

---

|            |                                               |
|------------|-----------------------------------------------|
| condformat | <i>Conditional formatting for data frames</i> |
|------------|-----------------------------------------------|

---

**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**

|   |                        |
|---|------------------------|
| x | A matrix or data.frame |
|---|------------------------|

**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 the 'RStudio Viewer' or an 'HTML browser', as available.

**Examples**

```

data(iris)
cf <- condformat(iris[1:5,])
## Not run:
print(cf)

## End(Not run)

cf <- condformat(iris[1:5,]) %>% rule_fill_gradient(Sepal.Length)
## Not run:
print(cf)

## End(Not run)

cf <- condformat(iris[1:5,]) %>%
  rule_fill_discrete(Sepal.Length, expression=Sepal.Width > 2)
## Not run:
print(cf)

## End(Not run)

```

---

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(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 | <i>Writes the table to an Excel workbook</i> |
|------------------|----------------------------------------------|

---

**Description**

Writes the table to an Excel workbook

**Usage**

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

**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 whole workbook file
- overwrite\_sheet     logical to overwrite the sheet

---

|                 |                                            |
|-----------------|--------------------------------------------|
| condformat2grob | <i>Converts the table to a grid object</i> |
|-----------------|--------------------------------------------|

---

**Description**

Converts the table to a grid object

**Usage**

```
condformat2grob(x, draw = TRUE)
```

**Arguments**

**x** A `condformat_tbl` object

**draw** A logical. If TRUE (default), the table is immediately drawn using `grid::draw()` and the grob is returned. If FALSE, the grob is returned without drawing. Set `draw=FALSE` when using the grob in composite images with `gridExtra::grid.arrange()` or `ggpubr::ggarrange()`.

**Value**

the grid object

**Examples**

```
library(condformat)
data.frame(Student = c("Alice", "Bob", "Charlie"),
            Evaluation = c("Great", "Well done", "Good job!")) %>%
  condformat() %>%
  condformat2grob()
```

---

|                 |                                                 |
|-----------------|-------------------------------------------------|
| 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)
cf <- condformat2html(condformat(iris[1:5,]))
## Not run:
print(cf)

## End(Not run)
```

---

|                  |                                         |
|------------------|-----------------------------------------|
| 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 |
|---|-------------------------|

**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                                    |
| ... | Deprecated: Arguments passed to htmlTable::htmlTableWidget |

**Value**

the htmlTable widget

**Examples**

```
## Not run:
data(iris)
cf <- condformat2widget(condformat(iris[1:5,]))
\dontrun{
print(cf)
}

## 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

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

## Arguments

|            |                                                                                                                                |
|------------|--------------------------------------------------------------------------------------------------------------------------------|
| x          | A condformat_tbl object                                                                                                        |
| ...        | Arguments passed on to <a href="#">htmltools::html_print</a>                                                                   |
| 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::htmlTable\(\)](#)

**Examples**

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

## End(Not run)
```

rule\_css

*Apply a CSS style property as a conditional formatting rule***Description**

Apply a CSS style property as a conditional formatting rule

**Usage**

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

**Arguments**

|            |                                                                                                                  |
|------------|------------------------------------------------------------------------------------------------------------------|
| x          | A condformat object, typically created with <code>condformat()</code>                                            |
| columns    | A character vector with column names to be coloured. Optionally <code>tidyselect::language()</code> 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**

```
data(iris)
cf <- condformat(iris[c(1:5, 51:55, 101:105),]) %>%
  rule_css(Species, expression = ifelse(Species == "setosa", "red", "darkgreen"),
           css_field = "color")
## Not run:
print(cf)

## End(Not run)
```

---

|               |                                                                   |
|---------------|-------------------------------------------------------------------|
| rule_fill_bar | <i>Fill column with a bar of a length proportional to a value</i> |
|---------------|-------------------------------------------------------------------|

---

## Description

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

## Usage

```
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 <code>condformat()</code>                                                                          |
| columns    | A character vector with column names to be coloured. Optionally <code>tidyselect::language()</code> 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                                                                                                                  |
| 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)
cf <- condformat(iris[c(1:5, 70:75, 120:125), ]) %>% rule_fill_bar("Sepal.Length")
## Not run:
print(cf)

## End(Not run)
```

---

|                    |                                         |
|--------------------|-----------------------------------------|
| 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(
  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 <code>condformat()</code>                                                                                             |
| columns    | A character vector with column names to be coloured. Optionally <code>tidyselect::language()</code> 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.                                                                 |
| 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\\_css\(\)](#), [rule\\_fill\\_bar\(\)](#), [rule\\_fill\\_gradient2\(\)](#), [rule\\_fill\\_gradient\(\)](#), [rule\\_text\\_bold\(\)](#), [rule\\_text\\_color\(\)](#)

**Examples**

```
data(iris)
cf <- 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"))

## Not run:
print(cf)

## End(Not run)

cf <- condformat(iris[c(1:5, 70:75, 120:125), ]) %>%
  rule_fill_discrete(c(starts_with("Sepal"), starts_with("Petal")),
                    expression = Sepal.Length > 4.6,
                    colours=c("TRUE"="red"))

## Not run:
print(cf)

## End(Not run)
```

---

|                    |                                                    |
|--------------------|----------------------------------------------------|
| rule_fill_gradient | <i>Fill column with sequential colour gradient</i> |
|--------------------|----------------------------------------------------|

---

**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 <code>condformat()</code>                                                                          |
| columns    | A character vector with column names to be coloured. Optionally <code>tidyselect::language()</code> 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.                                                         |

**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**

```
data(iris)
cf <- condformat(iris[c(1:5, 70:75, 120:125), ]) %>%
  rule_fill_gradient(Sepal.Length) %>%
  rule_fill_gradient(Species, expression=Sepal.Length - Sepal.Width)
## Not run:
print(cf)
```

```
## End(Not run)

cf <- 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)
## Not run:
print(cf)

## End(Not run)
```

---

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

```
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 <code>condformat()</code>                                                                                            |
| columns    | A character vector with column names to be colored. Optionally <code>tidyselect::language()</code> 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 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_css()`, `rule_fill_bar()`, `rule_fill_discrete()`, `rule_fill_gradient()`, `rule_text_bold()`, `rule_text_color()`

**Examples**

```
data(iris)
cf <- condformat(iris[c(1:5, 70:75, 120:125), ]) %>%
  rule_fill_gradient2(Sepal.Length) %>%
  rule_fill_gradient2(Species, expression=Sepal.Length - Sepal.Width)
## Not run:
print(cf)

## End(Not run)

cf <- 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)
## Not run:
print(cf)

## End(Not run)
```

---

|                             |                                            |
|-----------------------------|--------------------------------------------|
| <code>rule_text_bold</code> | <i>Use bold text if a condition is met</i> |
|-----------------------------|--------------------------------------------|

---

**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 <code>condformat()</code>                                            |
| columns    | A character vector with column names to be coloured. Optionally <code>tidyselect::language()</code> can be used. |
| expression | Condition that evaluates to TRUE for the rows where bold text should be applied.                                 |
| na.bold    | If TRUE, make missing values bold.                                                                               |
| 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_gradient2()`, `rule_fill_gradient()`, `rule_text_color()`

**Examples**

```
data(iris)
cf <- condformat(iris[c(1:5, 51:55, 101:105),]) %>%
  rule_text_bold(Species, expression = Species == "setosa")
## Not run:
print(cf)

## End(Not run)
```

---

|                 |                                                              |
|-----------------|--------------------------------------------------------------|
| rule_text_color | <i>Give a color to the text according to some expression</i> |
|-----------------|--------------------------------------------------------------|

---

**Description**

Give a color to the text according to some expression

**Usage**

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

**Arguments**

|            |                                                                                                                  |
|------------|------------------------------------------------------------------------------------------------------------------|
| x          | A condformat object, typically created with <code>condformat()</code>                                            |
| columns    | A character vector with column names to be coloured. Optionally <code>tidyselect::language()</code> can be used. |
| expression | Condition that evaluates to color names for the rows where text should be colored                                |
| 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\\_gradient2\(\)](#), [rule\\_fill\\_gradient\(\)](#), [rule\\_text\\_bold\(\)](#)

**Examples**

```
data(iris)
cf <- condformat(iris[c(1:5, 51:55, 101:105),]) %>%
  rule_text_color(Species, expression = ifelse(Species == "setosa", "blue", ""))
## Not run:
print(cf)

## End(Not run)
```

---

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

---

**Description**

Keeps the variables you mention in the printed table. Compared to [dplyr::select\(\)](#), `show_columns` does not remove the columns from the data frame, so formatting rules can still depend on them.

**Usage**

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

**Arguments**

|           |                                                                                                                                                                                      |
|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| x         | A condformat object, typically created with <a href="#">condformat()</a>                                                                                                             |
| columns   | A character vector with column names to be to show. It can also be an expression can be used that will be parsed according to <a href="#">tidyselect::language()</a> . See examples. |
| col_names | Character vector with the column names for the selected columns                                                                                                                      |

**Value**

The condformat object with the rule added

**See Also**

[dplyr::select\(\)](#)

**Examples**

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

# Include some columns:
cf <- condformat(x) %>% show_columns(c(Sepal.Length, Sepal.Width, Species))
## Not run:
print(cf)

## End(Not run)
cf <- condformat(x) %>% show_columns(c("Sepal.Length", "Sepal.Width", "Species"))
## Not run:
print(cf)

## End(Not run)

# Rename columns:
cf <- condformat(x) %>%
  show_columns(c(Sepal.Length, Species),
               col_names = c("Length", "Spec."))
## Not run:
print(cf)

## End(Not run)

# Exclude some columns:
cf <- condformat(x) %>% show_columns(c(-Petal.Length, -Petal.Width))
## Not run:
print(cf)

## End(Not run)

cf <- condformat(x) %>% show_columns(c(starts_with("Petal"), Species))
## Not run:
print(cf)

## End(Not run)

petal_width <- "Petal.Width"
cf <- condformat(x) %>% show_columns(!petal_width)
## Not run:
print(cf)

## End(Not run)
```

## Description

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

## Usage

```
show_rows(x, ...)
```

## Arguments

|                  |                                |
|------------------|--------------------------------|
| <code>x</code>   | condformat_tbl object          |
| <code>...</code> | Expressions used for filtering |

## Value

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

## See Also

[dplyr::filter\(\)](#)

## Examples

```
library(condformat)
data(iris)
x <- head(iris)
cf <- condformat(x) %>% show_rows(Sepal.Length > 4.5, Species == "setosa")
## Not run:
print(cf)

## End(Not run)
# Use it programatically
expr_as_text <- 'Sepal.Length > 4.5'
expr <- rlang::parse_expr(expr_as_text)
cf <- condformat(x) %>% show_rows(!! expr)
## Not run:
print(cf)

## End(Not run)
# With multiple arguments:
expr_as_text <- c('Sepal.Length > 4.5', 'Species == "setosa"')
exprs <- lapply(expr_as_text, rlang::parse_expr)
cf <- condformat(x) %>% show_rows(!!! exprs)
## Not run:
print(cf)

## End(Not run)
```

---

|               |                                                |
|---------------|------------------------------------------------|
| theme_caption | <i>Sets the caption of a condformat object</i> |
|---------------|------------------------------------------------|

---

**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)
cf <- condformat(head(iris)) %>%
  theme_caption(caption = "My Caption")
## Not run:
print(cf)

## End(Not run)
```

---

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

---

**Description**

This is only used on grob output.

**Usage**

```
theme_grob(x, ...)
```

**Arguments**

|     |                                                                            |
|-----|----------------------------------------------------------------------------|
| x   | The condformat object                                                      |
| ... | Arguments to be passed to <code>gridExtra::tableGrob</code> (see examples) |

**See Also**

[gridExtra::tableGrob\(\)](#)

**Examples**

```
data(iris)
cf <- condformat(head(iris)) %>%
  theme_grob(
    rows = NULL,
    theme = gridExtra::ttheme_default(base_size = 10, base_colour = "red")
  )
condformat2grob(cf)
```

---

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

---

**Description**

Customizes appearance of condformat object

**Usage**

```
theme_htmlTable(x, ...)
```

**Arguments**

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

**See Also**

[htmlTable::htmlTable\(\)](#)

**Examples**

```
data(iris)
cf <- condformat(head(iris)) %>%
  theme_htmlTable(caption="Table 1: My iris table", rnames=FALSE)
## Not run:
print(cf)

## End(Not run)
```

---

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

---

**Description**

Customizes appearance of condformat object

**Usage**

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

**Arguments**

|     |                                                                                  |
|-----|----------------------------------------------------------------------------------|
| x   | The condformat object                                                            |
| ... | Arguments to be passed to <code>htmlTable::htmlTableWidget</code> (see examples) |

**See Also**

[htmlTable::htmlTable\(\)](#)

**Examples**

```
data(iris)
cf <- condformat(head(iris)) %>%
  theme_htmlWidget(number_of_entries = c(10, 25, 100),
                    width = NULL, height = NULL, elementId = NULL)

## Not run:
print(cf)

## End(Not run)
```

---

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

---

**Description**

This is only used on LaTeX output.

**Usage**

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

**Arguments**

|     |                                                                    |
|-----|--------------------------------------------------------------------|
| x   | The condformat object                                              |
| ... | Arguments to be passed to <code>knitr::kable</code> (see examples) |

**See Also**[knitr::kable\(\)](#)**Examples**

```
data(iris)
cf <- condformat(head(iris)) %>%
  theme_kable(booktabs = TRUE, caption = "My Caption")
## Not run:
print(cf)

## End(Not run)
```

# Index

## \* rule

- rule\_css, 10
- rule\_fill\_bar, 11
- rule\_fill\_discrete, 12
- rule\_fill\_gradient, 13
- rule\_fill\_gradient2, 15
- rule\_text\_bold, 16
- rule\_text\_color, 17

  

- cf\_field\_to\_css, 2
- cf\_field\_to\_gtable, 3
- cf\_field\_to\_latex, 4
- condformat, 4
- condformat(), 10–12, 14, 15, 17, 18
- condformat-shiny, 5
- condformat2excel, 6
- condformat2grob, 6
- condformat2html, 7
- condformat2latex, 8
- condformat2widget, 8
- condformat\_example (condformat-shiny), 5
- condformatOutput (condformat-shiny), 5

  

- dplyr::filter(), 20
- dplyr::select(), 18

  

- gridExtra::grid.arrange(), 7
- gridExtra::tableGrob(), 21

  

- htmlTable::htmlTable(), 9, 22, 23
- htmltools::html\_print, 9

  

- knit\_print.condformat\_tbl, 9
- knitr::kable(), 24

  

- print.condformat\_tbl, 9

  

- renderCondformat (condformat-shiny), 5
- rule\_css, 10, 11, 13, 14, 16–18
- rule\_fill\_bar, 10, 11, 13, 14, 16–18
- rule\_fill\_discrete, 10, 11, 12, 14, 16–18
- rule\_fill\_gradient, 10, 11, 13, 13, 16–18
- rule\_fill\_gradient2, 10, 11, 13, 14, 15, 17, 18
- rule\_text\_bold, 10, 11, 13, 14, 16, 16, 18
- rule\_text\_color, 10, 11, 13, 14, 16, 17, 17

  

- show\_columns, 18
- show\_rows, 19

  

- theme\_caption, 21
- theme\_grob, 21
- theme\_htmlTable, 22
- theme\_htmlWidget, 23
- theme\_kable, 23
- tidyselect::language(), 10–12, 14, 15, 17, 18