Package 'condformat'

October 19, 2017

Type Package

Title Conditional Formatting in Data Frames

Version 0.7.0

Date 2017-10-19

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 htmlTable (>= 1.9), htmltools, tibble, scales, dplyr, lazyeval (>= 0.2.0), knitr, rmarkdown (>= 0.9.6), gplots, magrittr, rlang (>= 0.1.2), tidyselect

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 2017-10-19 20:00:34 UTC

R topics documented:

`+.condformat_tbl` *Combines data with formatting rules (deprecated)*

Description

This is deprecated

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` *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

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 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(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

```
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 workbook file
overwrite_sheet	logical to overwrite the sheet

---

condformat2html *Converts the table to a htmlTable object*

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 *Converts the table to LaTeX code*

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 *Converts the table to a htmlTableWidget*

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)
```

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
...	Provided for knitr_print compatibility

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_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 'condformat(x)'
`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_discrete, rule_fill_gradient2, rule_fill_gradient

Examples

```
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_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(...)

rule_fill_discrete_new(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

...	Dots are used to transition from the old syntax rule_fill_discrete_old to the new one
x	A condformat object, typically created with ‘condformat(x)‘
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.
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.

Details

The syntax in condformat rules has changed since v0.7. See rule_fill_discrete_old

Value

The condformat_tbl object, with the added formatting information

See Also

Other rule: rule_css, rule_fill_gradient2, 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"))

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"))
```

---

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

```
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_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

```
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(...)

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

Arguments

...	Dots are used to transition from the old syntax rule_fill_discrete_old to the new one
x	A condformat object, typically created with ‘condformat(x)‘
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.
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.

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_discrete, rule_fill_gradient2

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)

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 colour gradient*

Description

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

Usage

```
rule_fill_gradient2(...)

rule_fill_gradient2_new(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

...	Dots are used to transition from the old syntax rule_fill_discrete_old to the new one
x	A condformat object, typically created with 'condformat(x)'
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.
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.

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_discrete, 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)

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 colour gradient (deprecated)*

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.

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_gradient2_old

Fill column with divergent colour gradient (deprecated)

Description

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

Usage

```
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 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

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 = "#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.

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)
```

---

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.
`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

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)
```

---

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

```
show_columns(...)

show_columns_new(x, columns, col_names)
```

Arguments

`...`	Dots are used to transition from the old syntax `show_columns_old` to the new one
`x`	A condformat object, typically created with 'condformat(x)'
`columns`	A character vector with column names to be coloured. 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

Value

The condformat object with the rule added

See Also

`select`

Examples

```
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))

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

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

`show_columns_` *Show columns (deprecated)*

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 *Selects the variables to be printed (deprecated)*

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

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

```
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(...)

show_rows_new(x, ...)
```

Arguments

`...`	Expressions used for filtering
`x`	condformat_tbl object

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")
# Use it programatically
expr_as_text <- 'Sepal.Length > 4.5'
expr <- rlang::parse_quosure(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_quosure)
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.

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 *Customizes appearance of condformat object*

Description

Customizes appearance of condformat object

Usage

```
theme_htmlTable(...)

theme_htmlTable_new(x, ...)
```

Arguments

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

See Also

htmlTable

Examples

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

Index