Package 'colorDF'

July 1, 2020

Title Colorful Data Frames in R Terminal

Version 0.1.2

Description Colorful Data Frames in the terminal. The new class does change the behaviour of any of the objects, but adds a style definition and a print method. Using ANSI escape codes, it colors the terminal output of data frames. Some column types (such as p-values and identifiers) are automatically recognized.

Depends R (>= 2.10)

License GPL-3

Encoding UTF-8

LazyData true

Imports purrr,crayon

Suggests knitr, rmarkdown, fansi, htmltools, tidyverse, data.table, testthat (>= 2.1.0)

VignetteBuilder knitr

RoxygenNote 7.1.0

NeedsCompilation no

Author January Weiner [aut, cre] (<https://orcid.org/0000-0003-1438-7819>)

Maintainer January Weiner <january.weiner@gmail.com>

Repository CRAN

Date/Publication 2020-07-01 12:20:12 UTC

R topics documented:

colorDF-package 2
add_colorDF_theme 2
as.colorDF 3
colorDF 3
colorDF_themes 4
colorDF_themes_show 4

col_type<- 5
df_style<- 6
example_colorDF 7
format_col 8
get_colorDF_theme 8
highlight 9
print.colorDF 10
remove_df_style 11
summary.colorDF 11

Index **13**

`colorDF-package` *colorDF – colorful data frames in your terminal*

Description

colorDF – colorful data frames in your terminal

Details

colorDF allows you to view data frames using the color and styling capabilities of an ANSI terminal: 216 colors! 16 base colors! 24 shades of gray! Italic, bold, inverse *and* underline! Well, this may not seem much, but in fact it allows at least some basic highlighting or coloring significant p-values in red. Trust me, it is useful if you work a lot with huge data frames.

`add_colorDF_theme` *Add a new theme*

Description

Add a new theme

Usage

```
add_colorDF_theme(theme, id, description = NULL)
```

Arguments

`theme`	a list containing style definitions
`id`	an identifier (name) for the theme
`description`	Description of th theme

Value

invisibly the new theme.

Examples

```
newtheme <- get_colorDF_theme("bw")
## Like "bw" theme, but significant p-values are red
newtheme$data.styles$pval$fg_sign <- "#FF0000"
add_colorDF_theme(newtheme, "new", "My new theme")
```

`as.colorDF` *Make a dataframe colorful*

Description

Make a dataframe colorful

Usage

```
as.colorDF(x, ...)
```

Arguments

`x`	a data frame or similar object (e.g. tibble)
`...`	further arguments are passed to [colorDF()].

Value

a colorful data frame – identical object but with the ‘.style‘ attribute set.

See Also

[df_style()] on how to modify style of the colorful data frame

---

`colorDF` *Make a dataframe colorful*

Description

Make a dataframe colorful

Usage

```
colorDF(x, theme = NULL)
```

Arguments

`x`	a data frame or similar object (e.g. tibble)
`theme`	Which theme to use

Value

a colorful data frame – identical object but with the '.style' attribute set.

See Also

[colorDF_themes()] to list all themes; [colorDF_themes_show()] to view all themes.

[df_style()] on how to modify style of the colorful data frame

---

colorDF_themes *List all available themes for colorful data frames*

Description

List all available themes for colorful data frames

Usage

```
colorDF_themes()
```

Value

A character vector with the names of the themes

See Also

[colorDF_themes_show()] for a demonstration of all themes.

---

colorDF_themes_show *Demonstrate all defined themes*

Description

Demonstrate all defined themes

Usage

```
colorDF_themes_show(themes = NULL)
```

Arguments

themes	character vector with theme names to show

Details

"Themes" are simply predefined styles for colorful data frames. Some are suitable only for dark or light backgrounds, so this function is useful for choosing what looks best on your terminal.

When a colorful data frame is created with [colorDF()] or [as.colorDF()], the default theme is assigned to it. The default theme is defined by the option "colorDF_theme" set using [options()] (at startup, the default theme is "light").

You can also specify the theme to use when making a data frame colorful with [colorDF()] by using the 'theme=' parameter.

Examples

```
colorDF_themes_show()
colorDF_themes_show(themes=c("wb", "bw"))
```

---

col_type<- *Set or retrieve a column type*

Description

Set or retrieve a column type of a colorful data frame

Usage

```
col_type(x, cols = NULL) <- value

col_type(x, cols = NULL)
```

Arguments

x	a colorful data frame
cols	column names to set or retrieve
value	character vector with column types

Examples

```
mc <- colorDF(mtcars)
col_type(mc, "gear") <- "factor"
col_type(mc, "gear")
col_type(mc) <- list(gear="factor", cyl="integer")
```

`df_style<-` *Get or set style of a colorful data frame*

Description

Get or set style of a colorful data frame

Usage

```
df_style(x, element = NULL) <- value

df_style(x, element)
```

Arguments

`x`	a colorful data frame
`element`	element or elements of the style
`value`	one or more values to set

Details

Colorful data frames store the styles in the '.style' attribute of the data frame object. This attribute is a list with a number of keywords:

* fg, bg, decoration: formatting styles to be applied to the whole table (see "Formatting styles" below) * row.names, col.names, interleave: formatting styles for row names, table header and every second line in the table. If these elements are NULL, no styles will be applied. See "Formatting styles" below. * autoformat (logical): whether column type should be guessed from column names (which allows automatically recognizing a column called "pvalue" as a p-value and color significant cells. * col.styles: a list mapping the column names to formatting styles. * col.types: a list mapping the column names to column types. For example, if it is 'list(result="pval")', then the column with name "result" will be considered to be a p-value and styled accordingly. * type.styles: a list mapping column types to formatting styles. See "Column types" below. * fixed.width: if not NULL, all columns have the same width * sep: string separating the columns * digits: how many digits to use * tibble.style: if not NULL, cut off columns that do not fit the width

Value

'df_style(x)' returns a list. Assignment results in a data frame with a modified style.

Formatting styles

Each formatting style is a list describing style of the formatting and coloring the text elements. Following elements of that list are recognized:

* fg, bg: foreground and background colors specified as R name (use 'colors()' to get available colors) or HTML hexadicimal code * fg_sign: for p-values, foreground color for significant values * fg_true, fg_false: foreground colors for logical vectors * fg_neg: for numeric values, foreground

color for negative values * fg_na: color for NA values * is.pval: whether the values are to be treated as p-values * is.numeric: whether the values are to be treated as numeric * align: how the values should be aligned (right, left or center) * sign.thr: for p-values, the threshold of significance * digits: how many digits to use * decoration: a character vector which may include the following key words: inverse, bold, italic

Column types

Rather than directly assigning a style to a column (which is possible using the 'col.styles' element) it is preferable to change a style associated with a column type. Several such types are defined in the default styles:

* character * numeric * integer * factor * identifier * pval * default

See Also

[print.colorDF()] on printing options

Examples

```
df <- as.colorDF(mtcars)

## row names should be red on yellow background (yikes!)
df_style(df, "row.names") <- list(fg="red", bg="#FFFF00")

## example of assigning multiple values in one assignment:
df_style(df) <- list(interleave=list(fg="#FFFFFF", bg="blue"),
                  row.names=list(fg="blue", bg="#FFFF00"),
                  col.names=list(bg="#FFFFFF", fg="#FF00FF",
                                 decoration=c("bold", "italic")))
```

---

example_colorDF *Example data frame for colorDF*

Description

Example data frame for colorDF

format_col *Format a vector using styles*

Description

Format a vector (data frame column) aligning, rounding the numbers and adding color.

Usage

```
format_col(
  x,
  col_name = NULL,
  style = NULL,
  df_style = NULL,
  format = TRUE,
  col_width = NULL,
  prefix = " ",
  min.width = 5L
)
```

Arguments

x	a vector
col_name	optional: a column name (see Details)
style	A list with style definition
df_style	style for the whole data frame
format	Whether the vector should be formatted and aligned
col_width	optional: width to which elements of the vector should be aligned
prefix	prefix (column separator) to add to each element of x
min.width	minimum width of a column

get_colorDF_theme *Return a style defined in a theme*

Description

Return a style defined in a theme

Usage

```
get_colorDF_theme(theme)
```

Arguments

theme	name

Value

A list with the definitions of style

Examples

```
get_colorDF_theme("bw")
```

---

highlight *Highlight some rows in a data frame*

Description

Highlight some rows in a data frame

Usage

```
highlight(x, sel)
```

Arguments

x	data frame like object
sel	logical vector of length equal to number of rows in the data frame.

Details

Uses [print.colorDF()] to highlight selected rows in a data frame.

Examples

```
highlight(mtcars, mtcars$cyl == 6)
```

print.colorDF *Print method for colorful data frames*

Description

Print method for colorful data frames

Usage

```
## S3 method for class 'colorDF'
print(x, ...)

print_colorDF(
  x,
  n = getOption("colorDF_n"),
  width = getOption("width"),
  row.names = TRUE,
  tibble_style = getOption("colorDF_tibble_style"),
  highlight = NULL,
  sep = getOption("colorDF_sep"),
  ...
)
```

Arguments

x	a colorful data frame (object with class colorDF)
...	further arguments are ignored
n	Number of rows to show (default=20, use Inf to show all; this value can be set with options("colorDF_n"))
width	number of characters that the data frame should span on output
row.names	if TRUE (default), row names will be shown on output
tibble_style	whether to print with tibble style (overrides style setting)
highlight	a logical vector indicating which rows to highlight
sep	column separator string (overrides style setting)

Details

print_colorDF is the exported function, print.colorDF is the S3 method.

See Also

[df_style()] on how to modify colorful data frames

Examples

```
print(colorDF(mtcars))
print_colorDF(mtcars)
```

remove_df_style *Remove the colorful dataframe style attribute*

Description

Remove the colorful dataframe style attribute

Usage

```
remove_df_style(x)
```

Arguments

x	a colorful dataframe

Value

colorless data frame

summary.colorDF *Meaningful summary of data frames*

Description

Meaningful summary of data frames

Usage

```
## S3 method for class 'colorDF'
summary(object, ...)

summary_colorDF(object, ...)
```

Arguments

object	a data frame (possibly a color data frame)
...	ignored

Details

While this function is a summary method for objects of the colorDF class, it can also be applied to any other dataframe-like object.

'summary_colorDF' is the exported version of this function to facilitate usage in cases when converting an object to a colorDF is not desirable.

Value

A colorful data frame of class colorDF containing useful information on a dataframe-like object.

Examples

```
summary(colorDF(iris))
summary_colorDF(iris)
```

Index

add_colorDF_theme, 2
as.colorDF, 3

col_type *(*col_type<-*)*, 5
col_type<-, 5
colorDF, 3
colorDF-package, 2
colorDF_themes, 4
colorDF_themes_show, 4

df_style *(*df_style<-*)*, 6
df_style<-, 6

example_colorDF, 7

format_col, 8

get_colorDF_theme, 8

highlight, 9

print.colorDF, 10
print_colorDF *(*print.colorDF*)*, 10

remove_df_style, 11

summary.colorDF, 11
summary_colorDF *(*summary.colorDF*)*, 11