Package 'colorDF'

October 2, 2020

Title Colorful Data Frames in R Terminal

Version 0.1.4

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.

URL https://january3.github.io/colorDF/, https://github.com/january3/colorDF/

Depends R (>= 2.10)

License GPL-3

Encoding UTF-8

LazyData true

Imports purrr,crayon

Suggests knitr, rmarkdown, fansi, htmltools, data.table, testthat (>= 2.1.0), tibble, dplyr, tidyr

VignetteBuilder knitr

RoxygenNote 7.1.1

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 2020-10-02 14:52:06 UTC

R topics documented:

`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, OK, this may not seem much, but in fact it allows at highlighting, showing different column types or coloring significant p-values in red. Trust me, it is useful if you work a lot with huge data frames.

colorDF does not really introduce a new type of a data frame; when applied to tibbles, data frames or data tables it does not change their behavior except for the print method. In other words, it is only a visualization layer on top of a data frame like object, which will otherwise work exactly as expected.

To get started, continue with the `colorDF()` help page.

Known issues

- In Rstudio, inverse does not work correctly (known bug in Rstudio 1.3)
- colorDF relies on the crayon package. In certain instances, the crayon package will enforce only 8 basic colors even if more colors can be displayed. See the vignette of colorDF for an example and on how to deal with this issue.

See Also

colorDF() on creating colorful data frames; global options for colorDF; df_style() on how to modify style of the colorful data frame; col_type() on how to change column types; colorDF_themes() to list all themes; colorDF_themes_show() to view all themes.

---

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$type.styles$pval$fg_sign <- "#FF0000"
add_colorDF_theme(newtheme, "new", "My new theme")
```

---

colorDF *Make a dataframe colorful*

Description

Make a dataframe colorful

Usage

```
colorDF(x, theme = NULL)

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

Arguments

x	a data frame or similar object, e.g. tibble, data.table or any object for which as.data.frame call returns a data frame
theme	Which theme to use
...	further arguments are passed to colorDF().

Details

These functions turn any data frame like object (i.e. object which inherits the data.frame class, such as a tibble or a data table).

Apart from adding the S3 class "colorDF", the .style attribute (and later the .coltp attribute), the only thing that really changes is the print method (see print_colorDF()). In other words, the behavior of the object does not change (e.g., a base::data.frame() will by the default drop dimensions if one column is selected, while a tibble::tibble() will not). colorDF is just for visualization, never truly manipulation.

Several color themes come with the package; see colorDF_themes_show(). When creating a colorful data frame, a theme might be directly selected; otherwise the getOption("colorDF_theme") is consulted and if NULL, a default theme will be selected. The theme associated with an object becomes a style and can be further manipulated (see df_style()).

as.colorDF() calls colorDF(); this function is only here for completeness.

Value

a colorful data frame – identical object but with the .style attribute set and class "colorDF" added.

See Also

Introduction to the package; df_style() on how to modify style of the colorful data frame; col_type() on how to change column types; colorDF_themes() to list all themes; colorDF_themes_show() to view all themes.

Examples

```
colorDF(mtcars)
colorDF(mtcars, theme="bw")
```

---

colorDF-global-options

Global options for colorDF

Description

The behavior of colorful data frames can be influenced by a number of global options set with options(). All options and their defaults can be viewed with colorDF_options().

Usage

```
colorDF_options()
```

Details

The following global options are interpreted by functions in the colorDF package:

- `colorDF_n` (default: `20`): how many rows at maximum are printed (set to `Inf` to always show all rows).
- `colorDF_theme` (default: `"light"`): theme assigned by default to the new objects by `colorDF()` (and also when passing a data frame directly to `summary_colorDF()`).
- `colorDF_tibble_style` (default: `FALSE`): if `TRUE`, then only column will be shown which fit on the screen (much like in the default print method for tibbles.
- `colorDF_noitalic` (default: `FALSE`): some terminals do not support italics and instead use video inverse. This will make some styles look really weird. If this option is set to `TRUE` at time that the colorDF package is loaded, then the italic style will be silently ignored. Changing this option will have no effect when set after the package is loaded, so best put it in your `.Rprofile`.
- `colorDF_sep`: separator for the table columns
- `width`: width of the terminal in characters

See Also

`colorDF()` on creating colorful data frames; `df_style()` on how to modify style of the colorful data frame; `colorDF_themes()` to list all themes; `colorDF_themes_show()` to view all themes.

Examples

```
## use the dark theme for a terminal with dark background
options(colorDF_theme="dark")
colorDF(mtcars)
```

---

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, force_bg = FALSE)
```

Arguments

themes	character vector with theme names to show
force_bg	force background to "white" for light themes and "black" for dark themes

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

Details

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
- match
- hidden
- default

Of course, new column types may be defined and their formatting defined in a theme or a particular data frame style.

Examples

```
mc <- colorDF(mtcars)
col_type(mc, "gear") <- "factor"
col_type(mc, "gear")
col_type(mc) <- list(gear="factor", cyl="integer")
## Note: the *class* of the columns did not change!
```

```
## Chaning column type merely changes the way it is displayed
class(mc[["gear"]])

## Hide some columns
col_type(mc, c("disp", "hp")) <- "hidden"

## Create a new type and style
col_type(mc, "carb") <- "carbstyle"
df_style(mc)$type.styles$carbstyle <- list(fg="red", decoration="bold")
```

`df_search` *Search and highlight occurences of a pattern*

Description

Search and highlight occurences of a pattern in a data frame

Usage

```
df_search(x, pattern = NULL, cols = NULL)
```

Arguments

`x`	a data frame
`pattern`	a pattern; if NULL, the search results will be removed
`cols`	which columns to search for (if NULL, all columns will be searched)

Details

df_search is for highlighting cells matching a specific pattern.

Value

a color data frame object with the search pattern set for the given columns (or reset, if the pattern was NULL)

Examples

```
options(colorDF_tibble_style=TRUE)
if(require(dplyr)) {

  # Search for "blue" in any column
  starwars %>% df_search("blue")

  # Search in a specific column
  starwars %>% df_search("(Human|Wookie)", "species")

  # save the search pattern in a new object
  saved <- starwars %>% df_search("blue")
```

```
  # remove the search patterns
  saved <- df_search(saved)
}
```

`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 "Formatting styles" below and help page for `col_type()`.
- 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

See Also

`print.colorDF()` on printing options; `col_type()` for column types.

Examples

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

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

## you can use `$` to access the elements
## here, show significant p-values in green
df_style(df)$type.styles$pval$fg_sign <- "green"

## 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,
  max.width = NULL
)
```

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
max.width	maximum 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")

## use it to change the style of a colorDF
foo <- colorDF(mtcars)
df_style(foo) <- get_colorDF_theme("wb")
## Slightly modify the style
df_style(foo)$type.styles$pval$fg_sign <- "red"
```

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)
```

---

is.colorDF *Test whether an object has the class of colorDF*

Description

Test whether an object has the class of colorDF

Usage

```
is.colorDF(x)
```

Arguments

x	a data frame like object

Value

TRUE if the object is of colorDF class

---

print.colorDF *Print method for colorful data frames*

Description

This is the core of the colorDF package – a print method for the colorful (and other) 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"),
```

```
  bg = NULL,
  fg = NULL,
  ...
)
```

Arguments

x	a colorful data frame (object with class colorDF), a data.frame, a tibble, a data.table or any other object which can be coerced to data frame with as.data.frame function.
...	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)
bg	set default background for the table
fg	set default foreground for the table

Details

print_colorDF is the exported function, print.colorDF is the S3 method. Otherwise they are identical.

`print_colorDF` is a function that can be applied to any data frame like object. Using `colorDF()` to change the class of the object is only necessary if modifications to the style are required, such as specifying column type. However, `print_colorDF` applied to virtually any data frame, tibble or data table will work. In such a case, the theme used to display the data frame will either be taken from `getOption("colorDF_theme")` or a default theme will be used.

Column types

Column types are basically the column classes (numeric, factor etc.) plus a few specialized types (such as p-value) which are displayed slightly differently. For example, an identifier will usually be shown in bold, and significant p-values will be red (details depend on the given theme and style; see `col_type()` and `df_style()` for more information).

Changing the default methods

It is possible to assign `print_colorDF` to the default methods, thus changing the way how tibbles, data frames or other data frame like objects are displayed. This should be generally safe, but use it on your own peril and preferably only in interactive sessions. I use the following code in my .Rprofile file:

```
if(interactive()) {
  print.data.frame <- colorDF::print_colorDF
  print.tbl        <- colorDF::print_colorDF
  print.data.table <- colorDF::print_colorDF
}
```

See Also

df_style() on how to modify colorful data frames styles; col_type() on how to change the column types; colorDF_themes_show() to demonstrate available themes; highlight() and df_search() functions on how to use colorDF to highlight selected parts of a data frame.

Examples

```
colorDF(mtcars)
print_colorDF(mtcars, row.names=FALSE)

if(require(dplyr)) {
    starwars %>% colorDF
    starwars %>% print_colorDF(highlight=.$homeworld == "Tatooine")

    ## equivalently
    starwars %>% highlight(.$homeworld == "Tatooine")

    ## with another style
    options(colorDF_theme="bw")
    starwars %>% print_colorDF(tibble_style=TRUE, sep=" |%%| ")
}
```

---

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

Details

Strips the color data frame style, but leaves the .coltp and class intact.

Value

colorless data frame

See Also

To completely remove the colorDF class and attributes, use `uncolor()`

`summary_colorDF` *Meaningful summary of lists and data frames*

Description

Meaningful, row-wise summary function for lists and data frames

Usage

```
summary_colorDF(
  object,
  numformat = "quantiles",
  digits = 3,
  width = getOption("width")
)

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

Arguments

`object`	a data frame (possibly a color data frame)
`numformat`	format of the summary for numerical values. Can be one of "quantiles", "mean" and "graphics"
`digits`	number of significant digits to show (default: 3)
`width`	width of the summary table in characters
`...`	passed to `summary_colorDF`

Details

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

The summary table has five columns and as many rows as there are columns in the summarized data frame (or elements in a list). First four columns contain, respectively, column name, column class (abbreviated as in tibbles), number of unique values and number of missing values (`NA`'s). The contents of the fifth column depends on the column class and column type as follows:

- first, any lists are unlisted
- numeric columns (including integers) are summarized (see below)

- for character vectors and factors, if all values are unique or missing (NA) then this is stated explicitely
- otherwise, for character vectors and factors, the values will be listed, starting with the most frequent. The list will be shortened to fit the screen.

For numeric columns, by default the quantiles 0 (minimum), .25, .50 (median), .75 and 1 (maximum) are shown. Following alternatives can be specified using the option `numformat`:

- "mean": mean +- standard deviation
- "graphics": a graphical summary. Note that all numerical columns will be scaled with the same parameter, so this option makes sense only if the numerical columns are comparable. The graphics summary looks like this: —| + |—- and corresponds to a regular box plot, indicating the extremes and the three quartiles (- ... - indicates the data range, |...| the interquartile range and '+' stands for the median).

`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)
summary_colorDF(iris, numformat="g")
if(require(dplyr) && require(tidyr)) {
  starwars %>% summary_colorDF

  ## A summary of iris data by species
  iris %>%
    mutate(row=rep(1:50, 3)) %>%
    gather(key="parameter", value="Size", 1:4)  %>%
    mutate(pa.sp=paste(parameter, Species, sep=".")) %>%
    select(row, pa.sp, Size) %>%
    spread(key=pa.sp, value=Size) %>%
    select(-row) %>%
    summary_colorDF(numformat="g")
}
```

`term_boxplot` *Boxplot in a terminal*

Description

Show a boxplot using characters in the terminal window

Usage

```
term_boxplot(formula, data = NULL, width = getOption("width"))
```

Arguments

formula	a formula
data	data frame or matrix
width	width of the boxplot in characters

Value

invisibly return the color summary data frame used to draw the boxplot

Examples

```
term_boxplot(mpg ~ cyl, data=mtcars)
term_boxplot(Sepal.Length ~ Species, data=iris, width=70)
```

---

uncolor *Strip the colorDF class and style*

Description

Strip the colorDF class and style

Usage

```
uncolor(x)
```

Arguments

x	colorful data frame

Value

the original data frame like object

Index