Package ‘ggconf’

April 8, 2018

Type Package

Title Simpler Appearance Modification of 'ggplot2'

Version 0.1.3

Description A flexible interface for ggplot2::theme(), potentially saving 50% of your typing.

License GPL-3

Depends R (>= 3.3.0)

Imports ggplot2 (>= 2.2.0), rly (>= 1.3.0)

Suggests knitr, testthat, futile.logger (>= 1.4.3), dplyr, tibble

URL https://github.com/caprice-j/ggconf

BugReports https://github.com/caprice-j/ggconf/issues

Encoding UTF-8

LazyData true

RoxygenNote 6.0.1

VignetteBuilder knitr

Collate 'ggconf.R' 'prefix_match.R' 'approx_match.R' 'compiler.R'

NeedsCompilation no

Author Yasutaka Tanaka [aut, cre]

Maintainer Yasutaka Tanaka <tnk.yasutaka@gmail.com>

Repository CRAN

Date/Publication 2018-04-08 17:44:46 UTC

R topics documented:

`compile_ggconf` *the core function of ggconf*

Description

the core function of ggconf

Usage

```
compile_ggconf(cmd = "")
```

Arguments

`cmd`	preprocessed characters compile_ggconf returns a built theme() object as string.

`define_ggconf_constants` *define constant values used in ggconf*

Description

`define_ggconf_constants` has no side effect.

Usage

```
define_ggconf_constants()
```

Details

One thing to note is `define_ggconf_constants` set implicitly the preference order of geom_name in ggplot2. For example, 'a.txt' ambiguously matches to 'axis.text' and 'axis.title', but ggconf automatically uses 'axis.text' with or without a warning message about the ambiguity.

See Also

The preference order is used when doing partial match in GgplotParser.

exec_ggconf *execute raw ggconf commands*

Description

execute raw ggconf commands

Usage

```
exec_ggconf(raw_input = "", show_warn = TRUE, batch_mode = FALSE,
  as_string = FALSE, show_compiled = TRUE)
```

Arguments

raw_input	A ggconf command
show_warn	Whether to show a warning message when ambiguously matched. Default is TRUE.
batch_mode	Default is FALSE. If TRUE, the resulted ggplot object is returned.
as_string	Return the resulted ggplot2 object as a string not as a ggplot2 object. Default is FALSE.
show_compiled	Show the compiled ggplot2 executable command. Default is TRUE.

find_first_index *define constant values used in ggconf*

Description

define constant values used in ggconf

Usage

```
find_first_index(pattern = "sz", table = c("x", "y", "size", "shape",
  "colour", "fill", "alpha", "stroke"), show_warn = TRUE, debug = FALSE)
```

Arguments

pattern	the pattern string
table	a table of string to be matched
show_warn	boolean for showing ambiguous match warning
debug	show debugging message if true

`get_all_theme_aes` *get all theme element names*

Description

get all theme element names

Usage

```
get_all_theme_aes()
```

`get_analogue` *return resulted strings of approximate string match*

Description

return resulted strings of approximate string match

Usage

```
get_analogue(fuzzy_input = "axs.txt", possibilities = c("axis.text",
  "axis.text.x"), n_top = 5, case_sensitive = FALSE, cost = c(insertions =
  0.25, deletions = 3, substitutions = 2), threshold = 6, debug = FALSE)
```

Arguments

`fuzzy_input`	A character. Typically user input.
`possibilities`	A character vector one of which is assumed to be pointed by fuzzy_input.
`n_top`	An integer specifying the number of returned strings.
`case_sensitive`	Default is FALSE.
`cost`	A named vector
`threshold`	If costs are more than threshold, remove them from the result even if they are within top `n_top`. Default is 6.
`debug`	If true, it shows costs for top candidates. `get_analogue` is a key function for returning useful compile error message.

Examples

```
get_analogue("axis.txt", c("axis.text", "axis.text.x", "axis.ticks"))
# returns "axis.text" "axis.text.x" "axis.ticks"

get_analogue("p.bg", c("plot.background", "panel.background"))
# returns "plot.background" as first, and then "panel.background"
```

get_element_tree_clone

get element tree clone

Description

devtools::check() add a note about using ggplot2:::.element_tree because it is an internal object of other packages. Thus, a quick-and-dirty solution, I just copied the resulted data frame here.

Usage

```
get_element_tree_clone()
```

get_theme_elem_name_conf

get all theme element configurations

Description

get all theme element configurations

Usage

```
get_theme_elem_name_conf(class = "element_text")
```

Arguments

`class`	one of "element_text", "element_blank", "element_line", or "element_rect"

ggconf_dbgmsg

display ggconf debug message

Description

display ggconf debug message

Usage

```
ggconf_dbgmsg(...)
```

Arguments

`...`	a sequence of objects passed to message()

remove_element_whatever *remove element_* function calls*

Description

remove element_* function calls

Usage

```
remove_element_whatever(input = "theme(l=element_text(sz=20))")
```

Arguments

input	An input string passed from users

replace_marks *replace some marks for later ggconf parsing*

Description

replace some marks for later ggconf parsing

Usage

```
replace_marks(input = "theme2(l.txt(size=12))")
```

Arguments

input	An input string passed from users

show_fixit_diagnostics *Display useful debugging info for users*

Description

Display useful debugging info for users

Usage

```
show_fixit_diagnostics(err = list(id = "p_theme_elem:prefix_match", type =
  "Prefix match for theme element name failed.", input = "axis.tx:", elem_name =
  "axis.tx", elem_table = c("axis.text", "axis.title")))
```

Arguments

`err`	A list of error information

`theme2` *an enhanced version of ggplot2::theme()*

Description

theme2() has an enhanced version of ggplot2::theme() in terms of: 1. no element_(text|line|rect|grob|blank) specification 2. partial match for each configuration (e.g. size by sz)

Usage

```
theme2(...)
```

Arguments

`...`	theme element specification (see examples below)

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {

library(ggplot2)
ggplot(mtcars) + geom_point(aes(wt, hp, color=as.factor(cyl))) +
  theme2(
    text(f="bold", z=24, fmly="Times New Roman"),
    pnl.bg(fill="white"),
    lgd.box.margin(.2, .2, .2, .2, "cm"),
    lgd.box.bg(c="black"),
    lgd.key(fill="white"),
    lgd.position("bottom"),
    lgd.txt(z=rel(.8)),
    lgd.title(fmly="Consolas", c="royalblue"),
    axs.title(fmly="Consolas", c="royalblue"),
    axs.title.y(angle=0, vjust=.5),
    axs.txt(z=rel(1.1)),
    axs.line(arrow=arrow(type="open", angle=20), z=2),
    axs.tick(z=1),
    axs.tick.len(.5, "cm"),
    plt.subttl(f="plain", hjust=1),
    plt.margin(.3, .3, .3, .1, "inch")
  )


ggplot(mtcars) + geom_point(aes(wt, hp, color=cyl)) +
   theme2(a.txt(family = c("Consolas", "Times")[1]))
# all of the following three generate the same plot

ggplot(mtcars) + geom_point(aes(wt, hp, color=cyl)) +
   theme(text = element_text(size=20, face=”bold”),
   axis.line = element_line(size=2),
   legend.key = element_rect(color=”black”))

ggplot(mtcars) + geom_point(aes(wt, hp, color=cyl)) +
   theme2(text(size=20, face=”bold”), axis.line(size=2),
          legend.key(color=”black”))

ggplot(mtcars) + geom_point(aes(wt, hp, color=cyl)) +
   theme2(txt(sz=20, f=”bold”), aline(sz=2), l.key(c=”black”))



}
```

Index