Package: tantastic (via r-universe)

September 28, 2024

Title Fun and Shiny and Pretty Functions

Version 0.2.2

Description A personal package of R functions, ggplot themes, and other miscellany.

License MIT + file LICENSE

Depends R (>= 4.1.0)

Imports cli (>= 3.0.0), rlang, utils

Suggests DT (>= 0.15), ggplot2 (>= 3.3.0), extrafont (>= 0.15), purrr (>= 0.3.0), shiny, progressr, testthat (>= 3.0.0)

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

Config/testthat/edition 3

URL `https://github.com/tanho63/tantastic`

BugReports `https://github.com/tanho63/tantastic/issues`

Repository https://tanho63.r-universe.dev

RemoteUrl https://github.com/tanho63/tantastic

RemoteRef HEAD

RemoteSha f3ae3d0963a811a8af54ed93b49db56d89eb19c0

Contents

`coalesce_join` *Join two dataframes, coalescing any columns with common names*

Description

Join two dataframes, coalescing any columns with common names

Usage

```
coalesce_join(x, y, by, type = c("left", "inner", "full"))
```

Arguments

`x, y`	A pair of data frames, data frame extensions (e.g. a tibble), or lazy data frames (e.g. from dbplyr or dtplyr). See *Methods*, below, for more details.
`by`	A join specification created with `join_by()`, or a character vector of variables to join by. If `NULL`, the default, `*_join()` will perform a natural join, using all variables in common across `x` and `y`. A message lists the variables so that you can check they're correct; suppress the message by supplying by explicitly. To join on different variables between `x` and `y`, use a `join_by()` specification. For example, `join_by(a == b)` will match `x$a` to `y$b`. To join by multiple variables, use a `join_by()` specification with multiple expressions. For example, `join_by(a == b, c == d)` will match `x$a` to `y$b` and `x$c` to `y$d`. If the column names are the same between `x` and `y`, you can shorten this by listing only the variable names, like `join_by(a, c)`. `join_by()` can also be used to perform inequality, rolling, and overlap joins. See the documentation at ?join_by for details on these types of joins. For simple equality joins, you can alternatively specify a character vector of variable names to join by. For example, `by = c("a", "b")` joins `x$a` to `y$a` and `x$b` to `y$b`. If variable names differ between `x` and `y`, use a named character vector like `by = c("x_a" = "y_a", "x_b" = "y_b")`. To perform a cross-join, generating all combinations of `x` and `y`, see `cross_join()`.

`dt_fmt_col` *Colour DT column by range*

Description

Colours DT columns based on the minimum and maximum of each column. Defaults to a purple-green colour scheme HULK which is colourblind-friendly, and naturally associates "green" to "good" and "purple" to "bad".

Usage

```
dt_fmt_col(
  dt,
  columns,
  column_range = NULL,
  colours = c("#9162C5", "#F1F1F1", "#6BA359"),
  reverse_colours = FALSE,
  colour_steps = 100
)
```

Arguments

`dt`	a DT object created by `DT::datatable()`
`columns`	a character or numeric vector of column identifiers (column names or indices)
`column_range`	numeric or "auto": either the minimum and maximum range to be coloured, or "auto" which defaults to the max and min of the table column.
`colours`	A set of hex colours to be passed on to `colorRampPalette()`, defaults to purple-green
`reverse_colours`	logical: reverses direction of colour scale
`colour_steps`	number: how many distinct colours to create, default 100

Value

A DT object with added colour formatting

Examples

```
DT::datatable(mtcars) |>
  # standard: formats high values as green, low values as purple
  dt_fmt_col(columns = c("hp", "cyl")) |>
  # reverse: formats low values as green, high values as purple
  dt_fmt_col(columns = "mpg", reverse_colours = TRUE) |>
  # custom colours
  dt_fmt_col(columns = "wt", colours = c("orange", "white", "blue"))
```

gen_input_map *Generate Shiny Inputs for a vector of identifiers*

Description

Generate Shiny Inputs for a vector of identifiers

Usage

```
gen_input_map(uid, FUN, id_prefix = NULL, ...)
```

Arguments

uid	a vector of unique identifiers
FUN	a ShinyInput function - should theoretically work for all shinyInput functions that have `inputid` as the first argument
id_prefix	a string that will be combined with the unique identifier and passed as the inputid
...	other arguments to be passed to the `FUN`

Value

a character vector of shinyInputs

Examples

```
gen_input_map(1:5, shiny::numericInput, id_prefix = "playerid_", label = "my_label", value = 1)
```

git_cleanup *Delete merged GitHub branches*

Description

Delete merged GitHub branches

Usage

```
git_cleanup()
```

Value

invisible(TRUE)

`progressively` *Progressively*

Description

This function helps add progress-reporting to any function - given function `f()` and progressor `p()`, it will return a new function that calls `f()` and then (on-exiting) will call `p()` after every iteration. Now superseded by purrr's map `.progress` arguments.

Usage

```
progressively(f, p = NULL)
```

Arguments

`f`	a function to add progressr functionality to.
`p`	a progressor function similar to that created by `progressr::progressor()`

Details

This is inspired by purrr's `safely`, `quietly`, and `possibly` function decorators.

Value

a function that does the same as `f` but it calls `p()` after iteration.

Examples

```
try({
urls <- c("https://github.com/nflverse/nflverse-data/releases/download/test/combines.csv",
        "https://github.com/nflverse/nflverse-data/releases/download/test/combines.csv")
read_test_files <- function(urls){
  p <- progressr::progressor(along = urls)
  lapply(urls, progressively(read.csv, p))
}

progressr::with_progress(read_test_files(urls))
# superseded by
purrr::map(urls, read.csv, .progress = TRUE)
})
```

read_inputs *Read a sequence of Shiny Inputs*

Description

Read a sequence of Shiny Inputs

Usage

```
read_inputs(
  inputid = NULL,
  nullarg = NA,
  type = c("chr", "dbl", "lgl", "int"),
  .env = shiny::getDefaultReactiveDomain()
)
```

Arguments

`inputid`	a vector of inputids to read
`nullarg`	the value to return if the input value is `NULL` i.e. missing
`type`	one of (`'chr'`,`'dbl'`,`'lgl'`,`'int'`) - specifies type of atomic vector to return
`.env`	the environment to look for the input - defaults to the default reactive domain

Value

a vector of values

Examples

```
if(interactive()){
  read_inputs(inputid = c("select_1", "select_2"), nullarg = NA, type = "chr")
}
```

set_geom_colour_defaults *Update geom defaults*

Description

Update geom defaults

Usage

```
set_geom_colour_defaults(colour = "#57c1f1")
```

Arguments

`colour`	colour of geom default

`set_geom_font_defaults` *Update matching font defaults for text geoms*

Description

Updates ggplot2::geom_label and ggplot2::geom_text font defaults

Usage

```
set_geom_font_defaults(
  family = "IBM Plex Sans",
  face = "plain",
  size = 3.5,
  color = "#2b2b2b"
)
```

Arguments

`family`, `face`, `size`, `color`	font family name, face, size and color

`str` *Tan's str() function*

Description

Wraps `utils::str()` but defaults to max.level = 2

Usage

```
str(..., max.level = 2)
```

Arguments

`...`	objects passed to str
`max.level`	sets max.level - by default, 2

Value

output of utils::str() but defaults to max.level = 2

Examples

```
list(
  data = list(
    mtcars = data.frame(mtcars),
    airquality = data.frame(airquality)
  )
) |>
  str()
```

`theme_tantastic` *Tan's theme - dark*

Description

Largely inspired by hrbrthemes's modern rc with other stuff.

Usage

```
theme_tantastic(
  base_family = "IBM Plex Sans Condensed",
  base_size = 11.5,
  plot_title_family = "Bai Jamjuree",
  plot_title_size = 18,
  plot_title_face = "bold",
  plot_title_margin = 10,
  subtitle_family = base_family,
  subtitle_size = 14,
  subtitle_face = "plain",
  subtitle_margin = 15,
  strip_text_family = base_family,
  strip_text_size = 14,
  strip_text_face = "plain",
  caption_family = plot_title_family,
  caption_size = 14,
  caption_face = "plain",
  caption_margin = 14,
  axis_text_size = base_size,
  axis_title_family = base_family,
  axis_title_size = 12,
  axis_title_face = "plain",
  axis_title_just = "rt",
  plot_margin = ggplot2::margin(30, 30, 30, 30),
  grid = TRUE,
  axis = FALSE,
  ticks = FALSE
)
```

Arguments

`base_family`, `base_size`	base font family and size
`plot_title_family`, `plot_title_face`, `plot_title_size`, `plot_title_margin`	plot title family, face, size and margi
`subtitle_family`, `subtitle_face`, `subtitle_size`	plot subtitle family, face and size
`subtitle_margin`	plot subtitle margin bottom (single numeric value)
`strip_text_family`, `strip_text_face`, `strip_text_size`	facet label font family, face and size
`caption_family`, `caption_face`, `caption_size`, `caption_margin`	plot caption family, face, size and margin
`axis_text_size`	font size of axis text
`axis_title_family`, `axis_title_face`, `axis_title_size`	axis title font family, face and size
`axis_title_just`	axis title font justification, one of `[blmcrt]`
`plot_margin`	plot margin (specify with `ggplot2::margin()`)
`grid`	panel grid (`TRUE`, `FALSE`, or a combination of `X`, `x`, `Y`, `y`)
`axis`	add x or y axes? `TRUE`, `FALSE`, `"xy"`
`ticks`	ticks if `TRUE` add ticks

Examples

```
## Not run:
library(ggplot2)
library(dplyr)

# seminal scatterplot
ggplot(mtcars, aes(mpg, wt)) +
  geom_point() +
  labs(x="Fuel efficiency (mpg)", y="Weight (tons)",
       title="Seminal ggplot2 scatterplot example",
       subtitle="A plot that is only useful for demonstration purposes",
       caption="Brought to you by the letter 'g'") +
  theme_tantastic()

# seminal bar chart
count(mpg, class) |>
  ggplot(aes(class, n)) +
  geom_col() +
  geom_text(aes(label=n), nudge_y=3) +
  labs(x="Fuel efficiency (mpg)", y="Weight (tons)",
       title="Seminal ggplot2 bar chart example",
       subtitle="A plot that is only useful for demonstration purposes",
      caption="Brought to you by the letter 'g'") +
  theme_tantastic(grid="Y") +
  theme(axis.text.y=element_blank())

## End(Not run)
```

---

theme_uv *Tan's theme - dark*

Description

Largely inspired by hrbrthemes's modern rc with other stuff.

Usage

```
theme_uv(
  base_family = "IBM Plex Sans Condensed",
  base_size = 11.5,
  plot_title_family = "Bai Jamjuree",
  plot_title_size = 18,
  plot_title_face = "bold",
  plot_title_margin = 10,
  subtitle_family = base_family,
  subtitle_size = 14,
  subtitle_face = "plain",
  subtitle_margin = 15,
  strip_text_family = base_family,
  strip_text_size = 14,
  strip_text_face = "plain",
  caption_family = plot_title_family,
  caption_size = 14,
  caption_face = "plain",
  caption_margin = 14,
  axis_text_size = base_size,
  axis_title_family = base_family,
  axis_title_size = 12,
  axis_title_face = "plain",
  axis_title_just = "rt",
  plot_margin = ggplot2::margin(30, 30, 30, 30),
  grid = TRUE,
  axis = FALSE,
  ticks = FALSE
)
```

Arguments

`base_family, base_size`	base font family and size
`plot_title_family, plot_title_face, plot_title_size, plot_title_margin`	plot title family, face, size and margi
`subtitle_family, subtitle_face, subtitle_size`	plot subtitle family, face and size
`subtitle_margin`	plot subtitle margin bottom (single numeric value)
`strip_text_family, strip_text_face, strip_text_size`	facet label font family, face and size
`caption_family, caption_face, caption_size, caption_margin`	plot caption family, face, size and margin
`axis_text_size`	font size of axis text
`axis_title_family, axis_title_face, axis_title_size`	axis title font family, face and size
`axis_title_just`	axis title font justification, one of `[blmcrt]`
`plot_margin`	plot margin (specify with `ggplot2::margin()`)
`grid`	panel grid (`TRUE`, `FALSE`, or a combination of `X`, `x`, `Y`, `y`)
`axis`	add x or y axes? `TRUE`, `FALSE`, `"xy"`
`ticks`	ticks if `TRUE` add ticks

Examples

```
## Not run:
library(ggplot2)
library(dplyr)

# seminal scatterplot
ggplot(mtcars, aes(mpg, wt)) +
  geom_point() +
  labs(x="Fuel efficiency (mpg)", y="Weight (tons)",
       title="Seminal ggplot2 scatterplot example",
       subtitle="A plot that is only useful for demonstration purposes",
       caption="Brought to you by the letter 'g'") +
  theme_uv()

# seminal bar chart
count(mpg, class) |>
  ggplot(aes(class, n)) +
  geom_col() +
  geom_text(aes(label=n), nudge_y=3) +
  labs(x="Fuel efficiency (mpg)", y="Weight (tons)",
       title="Seminal ggplot2 bar chart example",
       subtitle="A plot that is only useful for demonstration purposes",
       caption="Brought to you by the letter 'g'") +
  theme_uv(grid="Y") +
  theme(axis.text.y=element_blank())

## End(Not run)
```

`unbind_dt` *Unbind Shiny Inputs (server function) When generating inputs reactively, it's sometimes necessary to unbind old inputs so that you can use the updated inputs in Shiny. '*`unbind_dt_js()` *adds the JS to the UI, while* `unbind_dt()` *is called as needed in the server component (usually as part of an* `observeEvent` *or* `eventReactive`*)*

Description

Unbind Shiny Inputs (server function) When generating inputs reactively, it's sometimes necessary to unbind old inputs so that you can use the updated inputs in Shiny. '`unbind_dt_js()` adds the JS to the UI, while `unbind_dt()` is called as needed in the server component (usually as part of an `observeEvent` or `eventReactive`)

Usage

```
unbind_dt(dt_name, session = shiny::getDefaultReactiveDomain())
```

Arguments

`dt_name`	String representing the table's output ID (i.e. to unbind in output$table_alpha, use "table_alpha" as the parameter)
`session`	a `shiny::session` object

Value

a call to the JS that unbinds `dt_name`

`unbind_dt_js` *Unbind Shiny Inputs (UI function)*

Description

When generating inputs reactively, it's sometimes necessary to unbind old inputs so that you can use the updated inputs in Shiny. '`unbind_dt_js()` adds the JS to the UI, while `unbind_dt()` is called as needed in the server component (usually as part of an `observeEvent` or `eventReactive`)

Usage

```
unbind_dt_js()
```

Value

Adds JS to HTML head

use_client_tz *Get User's Reported Time Zone*

Description

Uses some Javascript to pass the client's timezone to the Shiny session.

Usage

```
use_client_tz(inputId = "_client_tz")

get_client_tz(
  inputId = "_client_tz",
  session = shiny::getDefaultReactiveDomain()
)
```

Arguments

`inputId`	character string, name of shiny input. Defaults to "_client_tz". When used in `use_client_tz`, may need to be namespaced if called within a module.
`session`	a `shiny::session` object, defaults to `shiny::getDefaultReactiveDomain`

Value

`use_client_tz()`: HTML tags to be included in Shiny UI

timezone

Examples

```
if(interactive()){

shiny::shinyApp(
  ui = shiny::fluidPage(use_client_tz(), shiny::textOutput("time")),
  server = function(input, output, session) {
    # can be used outside of reactive context, if desired
    tz <- get_client_tz()
    time <- format(Sys.time(), format = "%x %X", tz = tz)
    output$time <- renderText(paste0(tz,": ", time))
  }
)
}
```

Index