Package 'tntpr'

March 21, 2024

Type Package

Title Data Analysis Tools Customized for TNTP

Version 1.0.2

Description An assortment of functions and templates customized to meet the needs of data analysts at the non-profit organization TNTP. Includes functions for branded colors and plots, credentials management, repository set-up, and other common analytic tasks.

License CC BY 4.0

URL https://github.com/tntp/tntpr, https://tntp.github.io/tntpr/

Depends R (>= 3.2)

Imports cli, dplyr (>= 0.8.3), extrafont, formattable, ggplot2 (>= 3.2.1), grDevices, grid, janitor, keyring, labelled, lubridate (>= 1.7.4), magrittr (>= 1.5), purrr (>= 0.3.3), readr, rlang, rstudioapi, scales, stringr (>= 1.4.0), tibble (>= 2.1.3), tidyr (>= 1.0.0), tidyselect

Suggests devtools, knitr, rmarkdown, testthat (>= 3.0.0), usethis, ggridges, ggalt, forcats, qualtRics, haven

VignetteBuilder knitr

Encoding UTF-8

LazyData true

RoxygenNote 7.3.1

Config/testthat/edition 3

NeedsCompilation no

Author Dustin Pashouwer [aut, cre],
Sam Firke [aut],
Shane Orr [aut],
Sam Talcott [aut]

Maintainer Dustin Pashouwer <dustin.pashouwer@tntp.org>

Repository CRAN

Date/Publication 2024-03-21 14:50:02 UTC

R topics documented:

.onAttach *Title*

Description

Title

Usage

```
.onAttach(libname, pkgname)
```

Arguments

libname	library name
pkgname	package name

bar_chart_counts *Bar chart of counts with TNTP polish*

Description

Takes a user supplied data frame and turns the designated column into an N bar chart (uses position dodge from ggplot2).

Usage

```
bar_chart_counts(
  df,
  var,
  group_var = NULL,
  labels = "n",
  var_color = "green",
  group_colors = NULL,
  title = NULL,
  var_label = NULL,
  digits = 1,
  font = "Halyard Display",
  font_size = 12
)
```

Arguments

df	the data.frame to be used in the bar chart
var	unquoted column name for variable to count
group_var	(optional) unquoted column name for group variable. If this is specified, you get a 2-variable clustered bar chart. If left blank, a single variable bar chart.
labels	should labels show the count ("n") or the percentage ("pct")?
var_color	color for non-grouped charts; set to TNTP green by default. For both this and group_colors, strings will be tried in tntp_colors automatically. So c("red", "green") will get you the official TNTP colors, while c("red", "brown") will get you base R red and blue.
group_colors	character vector of group colors, if a specific palette is desired
title	main chart title
var_label	label for x-axis
digits	integer indicating the number of decimal places to be used in percentages. In truncating, ties are rounded up, like in MS Excel, i.e., 10.5 and 11.5 become 11 and 12. This is *not* base R's default behavior.
font	font for chart text; Segoe UI by default
font_size	size for chart text; set to 12 by default

Value

A ggplot object

Examples

```
# An N bar chart by default
# All examples use font = "sans" to avoid triggering font warnings
mtcars |>
  bar_chart_counts(var   = cyl,
                   var_color = "orange",
                   title = "Number of mtcars by cylinder",
                   font  = "sans")

# Use a grouping variable with custom colors
mtcars |>
  bar_chart_counts(var       = cyl,
                   group_var = vs,
                   group_colors = c("orange", "navy"),
                   labels    = "pct",
                   title     = "% of V vs. Straight engines by # of cylinders",
                   font  = "sans")
```

`check_all_count` *Tabulate a range of check-all-that-apply response columns in a single table.*

Description

This function is to be run on columns treated with `check_all_recode()`.

Takes a data.frame and range of columns containing all answer choices to a check-all-that-apply question and tabulates the results. People who did not select any choices (i.e., they did not answer the question) are omitted from the denominator. For this to make sense, the question's choices should be MECE, or there should be an NA option.

This works with an "Other" open-response text field, which will be recoded to a binary variable with `check_all_recode`.

Usage

```
check_all_count(dat, ...)
```

Arguments

`dat`	a data.frame with survey data
`...`	unquoted column names containing the range of the answer choices. Can be specified individually, as a range, i.e., `q1_1:q1_5`, or using other helper functions from `dplyr::select()`.

Value

a data.frame with the tabulated results (n and

Examples

```
x <- data.frame( # 4th person didn't respond at all
  unrelated = 1:5,
  q1_1 = c("a", "a", "a", NA, NA),
  q1_2 = c("b", "b", NA, NA, NA),
  q1_3 = c(NA, NA, "c", NA, NA),
  q1_other = c(NA, "something else", NA, NA, "not any of these")
)
library(dplyr) # for the %>% pipe
x %>%
  check_all_recode(q1_1:q1_other) %>%
  check_all_count(q1_1:q1_other)

# You can use any of the dplyr::select() helpers to identify the columns:
x %>%
  check_all_recode(contains("q1")) %>%
  check_all_count(contains("q1"))
```

`check_all_recode` *Process a range of check-all-that-apply response columns for correct tabulation.*

Description

Some survey software returns check-all-that-apply response columns where missing values could indicate either that the respondent skipped the question entirely, or that they did not select that particular answer choice. To count the responses properly, the cases where a respondent did not check any of choices - i.e., they skipped the question - should not be counted in the denominator (assuming that the choices were completely exhaustive, or that there was an NA option).

This function takes a data.frame and range of columns containing all answer choices to a check-all-that-apply question and updates the columns in the data.frame to contain one of three values: 1 if the choice was selected; 0 if the respondent chose another option but not this one; or NA if the respondent skipped the question (i.e., they did not select any of the choices) and thus their response is truly missing.

It also takes the single text values in each column and adds them as a `label` attribute to each data.frame columns.

This function accomodates an open-response column, to get the correct denominator when some respondents have skipped all check variables but written something in. This passing over of the offered choices is an implicit rejection of them, not a "missing." Such a text variable will throw a warning - which may be okay - and will then be recoded into a binary 1/0 variable indicating a response. Such a text variable will be assigned the label "Other". Consider preserving the original respondent text values prior to this point as a separate column if needed.

`check_all_recode()` prepares the data.frame for a call to its sister function `check_all_count()`. The label attribute is accessed by this function.

Usage

```
check_all_recode(dat, ..., set_labels = TRUE)
```

Arguments

`dat`	a data.frame with survey data
`...`	unquoted variable names containing the answer choices. Can be specified as a range, i.e., `q1_1:q1_5` or using other helper functions from `dplyr::select()`.
`set_labels`	should the label attribute of the columns be over-written with the column text? Allow this to be TRUE unless there are currently label attributes you don't wish to overwrite.

Value

the original data.frame with the specified column range updated, and with label attributes on the questions.

Examples

```
x <- data.frame( # 4th person didn't respond at all
  unrelated = 1:5,
  q1_1 = c("a", "a", "a", NA, NA),
  q1_2 = c("b", "b", NA, NA, NA),
  q1_3 = c(NA, NA, "c", NA, NA),
  q1_other = c(NA, "something else", NA, NA, "not any of these")
)
library(dplyr) # for the %>% pipe
x %>%
  check_all_recode(q1_1:q1_other)

# You can use any of the dplyr::select() helpers to identify the columns:
x %>%
  check_all_recode(contains("q1"))
```

---

choose_text_color *Choose a text color given a background color*

Description

Choose a text color given a background color

Usage

```
choose_text_color(bg_color)
```

Arguments

bg_color	a color

Value

"black" or "white"

---

colors_tntp *TNTP colors*

Description

This list of colors has been superseded by the new brand colors and the new function `tntp_colors()`.

Usage

```
colors_tntp
```

Format

An object of class `character` of length 34.

Examples

```
tntp_colors()
```

---

colors_tntp_likert *Likert pallette*

Description

This likert palette has been superseded by the new brand colors and the new function `tntp_palette()`.

Usage

```
colors_tntp_likert
```

Format

An object of class `character` of length 7.

Examples

```
tntp_palette('likert_6')
```

---

colors_tntp_likert_orange_to_green *Likert orange to green pallette*

Description

This likert palette has been superseded by the new brand colors and the new functions `tntp_colors()` and `tntp_palette()`.

Usage

```
colors_tntp_likert_orange_to_green
```

Format

An object of class `character` of length 7.

Examples

```
tntp_palette('bg_6')
```

colors_tntp_palette *TNTP pallette*

Description

This list of colors has been superseded by the new brand colors and the new function `tntp_colors()`.

Usage

```
colors_tntp_palette
```

Format

An object of class `character` of length 16.

Examples

```
tntp_colors()
```

date_to_sy *Convert a date value into its school year.*

Description

Checks to see if a date is past the user-specified cutoff point for delineating school years, then maps to the appropriate year.

Usage

```
date_to_sy(date_var, last_day_of_sy = NULL)
```

Arguments

`date_var`	the date to convert. Can be a `Date` object or a string in the form 'YYYY-MM-DD' or 'MM/DD/YYYY'
`last_day_of_sy`	the cutoff date, after which a date is considered part of the following school year. The year of this argument does not matter. Defaults (noisily) to July 1st.

Value

Returns a character vector in the format of "2013 - 2014"

A character vector the same length as `date_var`

Examples

```
date_to_sy(as.Date("2014-05-05"), as.Date("2000-07-01"))
date_to_sy(as.Date("2014-07-05"), as.Date("2000-07-01"))
```

`factorize_df` *Convert all character vectors containing a set of values in a data.frame to factors.*

Description

This function examines each column in a data.frame; when it finds a column composed solely of the values provided to the `lvls` argument it updates them to be factor variables, with levels in the order provided.

This is an alternative to calling `dplyr::mutate_at` with `factor()` and identifying the specific variables you want to transform, if you have several repeated sets of responses.

Usage

```
factorize_df(dat, lvls, ignore.case = NULL)
```

Arguments

`dat`	data.frame with some factor variables stored as characters.
`lvls`	The factor levels in your variable(s), in order. If you have a question whose possible responses are a subset of another question's, don't use this function; manipulate the specific columns with `dplyr::mutate_at`.
`ignore.case`	Logical. If TRUE, will match without checking case, using the capitalization from the `lvls` parameter for the final output. If not provided, the function will provide a warning if it detects columns that would match without checking case but will NOT coerce them.

Value

a data.frame the same size as `dat`, with factorization completed in place.

Examples

```
teacher_survey |>
  factorize_df(lvls = c("Strongly Disagree", "Disagree", "Somewhat Disagree",
                        "Somewhat Agree", "Agree", "Strongly Agree"))

# prints warning due to case mismatches:
teacher_survey |>
  factorize_df(lvls = c("Strongly disagree", "Disagree", "Somewhat disagree",
                        "Somewhat agree", "Agree", "Strongly agree"))
```

`fake_county` *Fake teacher roster dataset from OpenSDP*

Description

The Fake County synthetic panel dataset contains approximately 40,000 records comprising four years of data with roughly 10,000 teachers per year. The dataset includes information about teacher demographics, teaching assignments, salary, credentials, experience, evaluation scores, and hiring and retention status. It also includes information about school types and average student characteristics for each school. There are no real teachers in the dataset, but it is based on real data. Fake County was developed as an offshoot of the Strategic Data Project's work on human capital diagnostics for school districts and state education departments, and can be used for teaching or collaboration. The data was synthesized using the R synthpop package.

Usage

```
fake_county
```

Format

A data frame with 39,339 rows and 38 variables:

tid double: Teacher ID

fake_data double: Record Is Simulated

school_year double: School Year

school_code double: School Code

school_name character: School Name

t_male double: Teacher Is Male

t_race_ethnicity double: Teacher Race/Ethnicity

t_job_area double: Teacher Assignment Type

t_salary double: Monthly Salary

t_nbpts double: Teacher Has National Board Certification

t_tenured double: Teacher Is Tenured

t_experience double: Years of Teaching Experience

t_fte double: Teacher's FTE Status

t_highest_degree double: Teacher's Highest Degree

t_licensed_stem double: Teacher Is Licensed In STEM Field

t_eval_obs double: Evaluation Summary Observation Score

t_eval_growth double: Evaluation Summary Student Growth Score

t_stay double: Teacher in Same School in Following Year

t_transfer double: Teacher in Different School in Following Year

t_leave double: Teacher Not Teaching in Fake County Schools in Following Year

t_novice double: Teacher Is Novice First-Year Teacher

t_new_hire double: Teacher Did Not Teach in Fake County in Prior Year

sch_elem double: School Is Elementary School

sch_middle double: School Is Middle School

sch_high double: School Is High School

sch_alternative double: School Is Alternative School

sch_regular double: School Is Regular School

sch_title_1 double: School Is Title 1 School

sch_magnet double: School Is Magnet School

sch_vocational double: School is Vocational School

sch_region double: School Region Code

sch_calendar_type double: School Calendar Type

sch_iep_pct double: School Special Education Student Share in 2012-15

sch_minority_pct double: School Minority Student Share in 2012-15

sch_frpl_pct double: School Free and Reduced Price Lunch Student Share in 2012-15

sch_ela_avg double: School ELA Test Score Average in 2012-15 (in standard deviations)

sch_math_avg double: School Math Test Score Average in 2012-15 (in standard deviations)

sch_enroll_2015 double: School Enrollment in 2015

Source

https://github.com/OpenSDP/fake-county, posted under a Creative Commons license.

`figureN` *Create sequential figure numbers*

Description

Create sequential figure numbers

Usage

```
figureN(x)
```

Arguments

x	character string description of the figure

Value

An atomic character vector prepended with a Figure number

Examples

```
figureN("Distribution of cars by cylinder count")
# Inline RMarkdown code: `r figureN("Distribution of cars by cylinder count")`

#
```

`get_usable_family` *Checks if a font family is usable and returns a usable font if not*

Description

Helper function. Checks if a given family value is available, and if not returns the default font family ("sans" or user provided)

Usage

```
get_usable_family(family, silent = FALSE, default_family = "sans")
```

Arguments

`family`	the font family to check as a character
`silent`	logical. If TRUE doesn't raise a warning if the font family is unavailable
`default_family`	defaults to "sans", but can be set to another fallback family.

Value

a character of a usable font family

`header_tntp` *Insert header_script_tntp.*

Description

Call this function from inside a .R file in RStudio to insert the standard TNTP header into your active script.

Usage

```
header_tntp()
```

Value

nothing

Examples

```
header_tntp()
```

`import_segoe_ui` *Import Segoe UI Condensed font for use in charts*

Description

This function will check if Segoe UI is already accessible in R and if not it will attempt to import it using the `extrafont` package

Usage

```
import_segoe_ui()
```

Value

nothing

Examples

```
import_segoe_ui()
```

`is_color` *Validate color inputs*

Description

Validate color inputs

Usage

```
is_color(x)
```

Arguments

x	a color

Value

TRUE if x can be interpreted as a color

labelled_to_factors *Convert all* labelled*-class columns to factors.*

Description

Deprecated. Use the `as_factor()` function from the `haven` package instead for the same functionality.

Takes a data.frame, checks for columns that are class `labelled` from the `haven` package, and converts them to factor class.

Usage

```
labelled_to_factors(labels_df)
```

Arguments

`labels_df`	a data.frame containing some columns of class labelled

Value

Returns a data.frame, the same size as `labels_df`

Examples

```
tntpr::fake_county |>
  haven::as_factor()
```

palette_names *Pallette names*

Description

This list of palette names has been superseded by the new brand colors and new functions `tntp_colors()` and `tntp_palette()`. To see all of the new brand palettes, use `show_tntp_palette()`.

Usage

```
palette_names
```

Format

An object of class `character` of length 7.

Examples

```
show_tntp_palette()
```

`palette_tntp` *TNTP branded color palettes*

Description

This function as been superseded by `tntp_colors()` which has improved functionality and includes the most recent TNTP brand colors.

This function creates user defined color palette combinations for up to eleven colors. There are nine TNTP approved colors: dark_blue, medium_blue, light_blue, green, orange, gold, dark_grey (dark_gray), medium_grey (medium_gray), light_grey (light_gray). White and black are also available.

Usage

```
palette_tntp(...)
```

Arguments

`...`	supply quoted color names to include in color palette

Value

a character vector

Examples

```
library(ggplot2)

pal1_tntp <- tntp_colors("green", "gold", "orange")
pal2_tntp <- tntp_colors("navy", "cerulean", "sky")

p <- ggplot(mtcars, aes(wt, mpg))
p <- p + geom_point(aes(colour = factor(cyl)))
p

# Change colors to created palette
p <- p + scale_color_manual(values = pal1_tntp)
p

g <- ggplot(mtcars, aes(factor(cyl), mean(mpg)))
g <- g + geom_bar(aes(fill = factor(cyl)), stat = "identity")
g

# Change fill to created palette
g <- g + scale_fill_manual(values = pal2_tntp)
g
```

palette_tntp_scales *scale_palette_tntp*

Description

This function has been superseded by `tntp_palette()` which includes the new brand colors.

Usage

```
palette_tntp_scales(palette = palette_names)
```

Arguments

palette	the palette

Value

a character vector

Examples

```
colors <- tntp_palette("likert_5")
```

parse_date *Attempt to parse a date with common formats*

Description

Helper function for date_to_sy. Returns a date object as is, or noisily attempts to parse a string in the form YYYY-MM-DD or MM/DD/YYYY. If the date cannot be parsed, throws an error.

Usage

```
parse_date(date)
```

Arguments

date	a character or Date vector to parse

Value

a Date vector, the same length as ‘date‘

`prop_matching` *Calculate the percent of non-missing values in a character vector containing the values of interest. This is a helper function for factorize_df().*

Description

Calculate the percent of non-missing values in a character vector containing the values of interest. This is a helper function for factorize_df().

Usage

```
prop_matching(vec, valid_strings, ignore.case = FALSE)
```

Arguments

`vec`	character vector.
`valid_strings`	the values that the variable can possibly take on.
`ignore.case`	if TRUE, ignores case in matching

Value

a numeric proportion between 0 and 1.

`recode_to_binary` *Recode a variable into binary groups, e.g., "Top-2" and "Not in Top-2".*

Description

Recodes a character variable into a binary result, a two-level factor. All values matching of the supplied character strings in the `to_match` vector are coded into the first level of the factor; all other values are coded into the other level. `NA` remains `NA`. The default factor labels are "Selected" and "Not selected" but these can be overridden.

This recoding is not case-sensitive; if you specify "agree" as a top-2 value, "Agree" will be counted as Top-2, and vice versa.

Usage

```
recode_to_binary(
  x,
  to_match = c("strongly agree", "agree"),
  label_matched = "Selected",
  label_unmatched = "Not selected"
)
```

Arguments

x	the character or factor vector to be recoded
to_match	a character vector with the strings that should be put in the first level of the factor. Defaults to "strongly agree" and "agree" but can be overwritten.
label_matched	what should be the factor label of values that match the strings specified in to_match? Defaults to "Selected"
label_unmatched	what should be the factor label of values that don't match the strings specified in to_match? Defaults to "Not selected".

Value

a factor variable (for nicer ordering in calls to janitor::tabyl) with values mapped to the two levels.

Examples

```
agreement <- c(
  "Strongly agree", "Agree", "Somewhat agree",
  "Somewhat disagree", "Strongly disagree", "Frogs", NA
)

recode_to_binary(agreement) # default values of "strongly agree" and "agree" are used for recoding
recode_to_binary(agreement,
  label_matched = "Top-2 Agree",
  label_unmatched = "Not in Top-2"
) # custom labels of factor levels
recode_to_binary(agreement, "frogs")
recode_to_binary(
  agreement,
  "frogs",
  "FROGS!!!",
  "not frogs"
) # custom matching values & labels of factor levels

freq <- c("always", "often", "sometimes", "never")
recode_to_binary(freq, "always", "always", "less than always")
```

---

scale_colour_tntp *scale_color_tntp/scale_fill_tntp*

Description

These functions are deprecated. Please use scale_color_manual(values = tntp_palette(palette_name)) or scale_fill_manual(values = tntp_palette(palette_name)) instead.

Usage

```
scale_colour_tntp(palette = palette_names, ...)

scale_color_tntp(palette = palette_names, ...)

scale_fill_tntp(palette = palette_names, ...)
```

Arguments

palette	character string describing the desired palette from
...	other arguments to pass through to ggplot2::discrete_scale()

Value

a ggplot Scale object

Examples

```
library(ggplot2)
library(dplyr)

x <- mtcars %>%
  count(cyl, am) %>%
  mutate(am = as.factor(am))

ggplot(x, aes(x = cyl, y = n, fill = am)) + # you need a fill aesthetic
  geom_col() +
  scale_fill_manual(values = tntp_palette())
```

---

setup_repo *Initialize a new repository, and a single subfolder, TNTP style.*

Description

Create a new repository on Bitbucket, then set your working directory to that folder and run this function. It will set up the main repo folder as well as a single subfolder in which you can work on your immediate project.

You must specify the subfolder name as well as the long name associated with that project and the analyst(s) working on it. These latter two values are used to create a README.Md file.

Usage

```
setup_repo(project_path, subfolder, proj_name, analyst_name)
```

Arguments

project_path	the path to the main project directory. To use the current project, use 'project_path = here::here()'.
subfolder	a character vector containing the concise name of a project subfolder. E.g., if the repository is the name of a city "Anywhere City", a project subfolder might be "ela_access" or "aps_talent_landscape").
proj_name	the longer, full name of the subfolder project. This will appear in the subfolder's README.md file. E.g., "Access to Grade-Level ELA Content Pilot."
analyst_name	the name(s) of the analysts currently working on the subfolder project. This will appear in the subfolder's README.md file.

Value

nothing

Examples

```
# Setting up in a temporary directory
setup_repo(project_path = tempdir(),
           subfolder = "ela_access",
           proj_name = "Access to Grade-Level ELA Content",
           analyst_name = "Dustin Pashouwer and Sam Firke")
```

---

setup_subdirectory *Initialize a new subdirectory in an existing repository, TNTP style.*

Description

A repository might represent a region, like "Anywhere City", or a major client or contract, like "Midwestern Charter Network. Within that repo you would have a subfolder for each analysis project. This function creates such a subfolder and populates it with folders and a README.

To use: within an existing repository on Bitbucket, set your your working directory to that folder and run this function to create a sub-folder.

Use `setup_repo()` in a blank new repository to add the first project subfolder and create the RProject and .gitignore files. Add subsequent analysis project folders with this function.

Usage

```
setup_subdirectory(project_path, subfolder, proj_name, analyst_name)
```

Arguments

project_path	the path to the main project directory. To use the current project, use 'project_path = here::here()'.
subfolder	a character vector containing the concise name of a project subfolder. E.g., if the repository is the name of a city "Anywhere City", a project subfolder might be "ela_access" or "aps_talent_landscape").
proj_name	the longer, full name of the subfolder project. This will appear in the subfolder's README.md file.
analyst_name	the name(s) of the analysts currently working on the subfolder project. This will appear in the subfolder's README.md file.

Value

nothing

Examples

```
# Setting up in a temporary directory
setup_subdirectory(tempdir(),
                   subfolder = "ela_access",
                   proj_name = "Equitable Access to Grade-Level ELA",
                   analyst_name = "Dustin Pashouwer and Sam Firke")
```

---

set_data_memo_formatting

Set the formatting options for a TNTP Data Memo

Description

internal function that calls standard formatting options for the Data Memo RMarkdown template moved here to keep the actual memo template cleaner and easier to use

Usage

```
set_data_memo_formatting()
```

Value

nothing

Examples

```
set_data_memo_formatting()
```

show_in_excel *Write Dataframe to a temp excel file and open it.*

Description

Write Dataframe to a temp excel file and open it.

Usage

```
show_in_excel(.data)
```

Arguments

.data	Dataframe

Value

nothing

Examples

```
# View a data set in excel
mtcars |> show_in_excel()
```

standardize_case *Update case of a character vector*

Description

Helper function for factorize_df(). Returns a vector of the same length as vec, with any values that match values in valid_strings updated to the case in valid_strings

Usage

```
standardize_case(vec, new_case)
```

Arguments

vec	The character vector you want to update
new_case	A character vector of correctly cased strings

Value

a character vector the same length as vec

tableN *Create sequential table numbers*

Description

Create sequential table numbers

Usage

```
tableN(x)
```

Arguments

x	character string description of the figure

Value

An atomic character vector prepended with a Table number

Examples

```
tableN("Distribution of cars by cylinder count")
# Inline RMarkdown code: `r tableN("Distribution of cars by cylinder count")`
```

teacher_survey *Teacher survey data*

Description

Simulated teacher survey data. Data only includes teh four TNTP high expectations questions.

Usage

```
teacher_survey
```

Format

'teacher survey' A data frame with 5 columns and 20 rows. The five columns are a 'timing' column, followed by four column for each of the four high expectations questions. Responses are on the 'strongly agree' to strongly disagree' 6-point scale.

Source

simulated in 'data-raw/teacher_survey.R'

theme_tntp *TNTP's ggplot2 theme*

Description

This theme is superseded by [tntp_style()]. Ggplot2 theme customized for TNTP aesthetics

Usage

```
theme_tntp(
  show_legend_title = TRUE,
  base_size = 12,
  base_family = "Segoe UI",
  grid_color = "grey93",
  title_align = "center",
  title_color = "black",
  title_size = 12,
  subtitle_align = "center",
  subtitle_color = "black",
  subtitle_size = 12,
  caption_align = "right",
  caption_color = "black",
  caption_size = 12
)
```

Arguments

show_legend_title	logical. Should the legend title be shown? Leave as TRUE if you want to change the legend title with a subsequent line + labs(...).
base_size	base font size
base_family	base font family
grid_color	color for major gridlines
title_align	alignment of main title, defaults to "center"; also accepts "left" or "right"
title_color	color of title text
title_size	size of title text
subtitle_align	alignment of sub-title, defaults to "center"; also accepts "left" or "right"
subtitle_color	color of subtitle text
subtitle_size	size of subtitle text
caption_align	alignment of caption, defaults to "right"; also accepts "left" or "center"
caption_color	color of caption text
caption_size	size of caption text

Value

a ggplot theme object.

`theme_tntp_2018` *A precise & pristine ggplot2 theme with opinionated defaults and an emphasis on typography*

Description

This theme is superseded by `tntp_style()`.

Usage

```
theme_tntp_2018(
  base_family = "Segoe UI",
  base_size = 11.5,
  plot_title_family = base_family,
  plot_title_size = 18,
  plot_title_face = "bold",
  plot_title_margin = 10,
  subtitle_family = base_family,
  subtitle_size = 12,
  subtitle_face = "plain",
  subtitle_margin = 15,
  strip_text_family = base_family,
  strip_text_size = 12,
  strip_text_face = "plain",
  caption_family = base_family,
  caption_size = 9,
  caption_face = "italic",
  caption_margin = 10,
  axis_text = TRUE,
  axis_text_size = base_size,
  axis_title_family = subtitle_family,
  axis_title_size = 9,
  axis_title_face = "plain",
  axis_title_just = "rt",
  plot_margin = ggplot2::margin(30, 30, 30, 30),
  grid_col = "grey93",
  grid = TRUE,
  axis_col = "#cccccc",
  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`
: add x or y axes text? `X`, `Y`

`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_col, axis_col`
: grid & axis colors; both default to `#cccccc`

`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

Value

a ggplot theme object.

Building upon `theme_tntp`

The function is setup in such a way that you can customize your own one by just wrapping the call and changing the parameters. See source for examples.

Gotchas

There are distinctions between font names and various devices. Names that work for display graphics devices and bitmap ones such as png may not work well for PostScript or PDF ones. You may need two versions of a font-based theme function for them to work in a particular situation. This situation usually only arises when using a newer font with many weights but somewhat irregular internal font name patterns.

There is an option `hrbrthemes.loadfonts` which – if set to `TRUE` – will call `extrafont::loadfonts()` to register non-core fonts with R PDF & PostScript devices. If you are running under Windows, the package calls the same function to register non-core fonts with the Windows graphics device.

Examples

```
library(ggplot2)
library(dplyr)

# seminal scatterplot
ggplot(mtcars, aes(mpg, wt)) +
  geom_point() +
  labs(
    x = "Fuel effiiency (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'"
  ) +
  tntp_style(family = 'sans')

# 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'"
  ) +
  tntp_style(family = 'sans') +
  theme(axis.text.y = element_blank())
```

`tntpr` *Data Analysis Tools Customized for TNTP*

Description

An in-house TNTP R package. Includes tools for data manipulation, analysis, and reporting, including making TNTP-themed charts and documents. By and for TNTP data-using staff, though available to the broader public.

Author(s)

Maintainer: Dustin Pashouwer <dustin.pashouwer@tntp.org>

Authors:

- Sam Firke
- Shane Orr <shane.orr@tntp.org>
- Sam Talcott <sam.talcott@tntp.org>

See Also

Useful links:

- https://github.com/tntp/tntpr
- https://tntp.github.io/tntpr/

`tntp_colors` *TNTP Brand Colors*

Description

Translate human friendly TNTP brand color names like "medium_blue" into accurate hex values for use in plotting. This function can also be used to show a named vector of all available TNTP brand colors and values. Use `show_tntp_colors()` to quickly visualize selected colors in the plot window. For often used palettes of TNTP colors, see `tntp_palette()`.

Usage

```
tntp_colors(...)

show_tntp_colors(
  ...,
  pattern = NULL,
  labels = TRUE,
  borders = NULL,
  cex_label = 1,
  ncol = NULL
)
```

Arguments

`...`	Supply quoted TNTP color names to return. If no colors are specified, returns all available colors.
`pattern`	Optional regular expression. If provided, will return only brand colors that match the regular expression
`labels`	Logical. Label colors with names and hex values?
`borders`	Border color for each tile. Default uses `par("fg")`. Use `border = NA` to omit borders.
`cex_label`	Size of printed labels, as multiplier of default size.
`ncol`	Number of columns. If not supplied, tries to be as square as possible.

Value

- `tntp_colors()` returns a character vector of color codes
- `show_tntp_colors()` returns nothing

Examples

```
library(ggplot2)

# Use tntp_colors() to retrieve a single color...
ggplot(mtcars, aes(wt, mpg)) +
  geom_point(color = tntp_colors('green'))

#... multiple colors ...
ggplot(iris, aes(Sepal.Length, Sepal.Width, color = Species)) +
  geom_point() +
  scale_color_manual(values = tntp_colors('green', 'navy', 'red'))

#... or a list of all possible TNTP brand colors
tntp_colors()

# Use show_tntp_colors() to quickly see brand colors in the plotting window
show_tntp_colors('mint', 'moss', 'green')

# You can also use a pattern to return similar colors
show_tntp_colors(pattern = 'green')

# You can see all colors (and names) by running it with no arguments
show_tntp_colors()
```

---

tntp_cred *TNTP Credential Get/Set Command*

Description

A wrapper around the keyring package for secure credential management.

`tntp_cred()` will attempt to get a credential, and if no credential is found it will prompt you to add it (and then return it).

`tntp_cred_set()` will set a credential. By default it will prompt before overwriting any current credentials.

`tntp_cred_list()` will list all current credentials by sorted by service and username.

Usage

```
tntp_cred(service, username = NULL, keyring = NULL, prompt = NULL)

tntp_cred_set(
  service = NULL,
  username = NULL,
  keyring = NULL,
  prompt = NULL,
  overwrite = NULL
```

```
)

tntp_cred_list(service = NULL, keyring = NULL)
```

Arguments

service	The identifier for the credential you are pulling or setting
username	OPTIONAL. Can be used to specify different usernames for the same service
keyring	OPTIONAL. Can be used to specify a specific keyring
prompt	OPTIONAL. What text should be displayed above the input box for the key while setting?
overwrite	OPTIONAL. By default, `tntp_cred_set()` will prompt if it finds a credential already saved. Set this to `TRUE` to overwrite without prompting or `FALSE` to throw an error if a current credential is found.

Value

- `tntp_cred()` returns a stored (or newly created) credential
- `tntp_cred_set()` returns nothing
- `tntp_cred_list()` returns a 2-column data frame of services and usernames

Examples

```
# Using tntp_cred() with qualtRics
library(qualtRics)

# If no credential is set, this command will prompt for it first
qualtrics_token <- tntp_cred("QUALTRICS_TOKEN")
qualtrics_api_credentials(api_key = qualtrics_token,
                          base_url = 'tntp.co1.qualtrics.com')

# To overwrite your Qualtrics credential
tntp_cred("QUALTRICS_TOKEN", .set = TRUE)
```

---

tntp_palette *Common TNTP Color Palettes*

Description

Use or see

Usage

```
tntp_palette(palette = "likert_6", reverse = FALSE)

show_tntp_palette(..., reverse = FALSE, pattern = NULL)
```

Arguments

palette	Name of the TNTP palette you want to use. To see all available palettes, use show_tntp_palette()
reverse	Logical. If set to TRUE, reverses the direction of the palette.
...	Supply quoted TNTP palette names to visualize. If no names are specified, shows all available palettes.
pattern	Optional regular expression. If provided, will return only palettes that match the regular expression

Value

- tntp_palette() returns a character vector of color codes
- show_tntp_palette() returns nothing

Examples

```
library(ggplot2)

# Use to add a common palette to a ggplot visualization
ggplot(diamonds, aes(y = color, fill = cut)) +
  geom_bar(position = "fill") +
  scale_fill_manual(values = tntp_palette('blues', reverse = TRUE))


# Use show_tntp_palette() to visualize a single or multiple palettes
show_tntp_palette('likert_7')
show_tntp_palette('bg_5', 'likert_5')

# You can use a pattern to show similar palettes
show_tntp_palette(pattern = 'top2')
show_tntp_palette(pattern = '_6')

# Or run it with no specified palettes to see all available palettes
show_tntp_palette()

# For creating a continuous color palette, use scale_color_gradient()
# along with tntp_colors():
ggplot(mtcars, aes(hp, disp, color = mpg)) +
  geom_point(size = 3) +
  scale_color_gradient(low = tntp_colors('red'),
                       high = tntp_colors('green'))
```

---

tntp_style *Create TNTP themed ggplot2 charts*

Description

A custom theme including TNTP fonts and other defaults for styling ggplot2 charts.

Usage

```
tntp_style(
  family = "Halyard Display",
  header_family = family,
  base_size = 28,
  text_color = "#222222",
  caption_color = "#7D7E81",
  show_legend_title = FALSE,
  show_axis_titles = FALSE,
  grid = FALSE,
  grid_color = "#CBCBCB",
  title_align = "left",
  legend_align = "left",
  caption_align = "right"
)
```

Arguments

`family`	Base font family. Defaults to "Halyard Display".
`header_family`	Font family for title and subtitle. Defaults to the base font family.
`base_size`	Base font size. Recommended minimum value of 15.
`text_color`	Text color for titles, axes, legends, and facets.
`caption_color`	Text color for caption.
`show_legend_title`	Logical. Should the legend title be shown? Leave as `TRUE` if you want to change the legend title with a subsequent line `+ labs(...)`.
`show_axis_titles`	Which axis titles should be shown? Use `TRUE` or `FALSE` for toggle both titles, or `x` or `y` to show just that axis title.
`grid`	Which grid lines should be shown? Use `TRUE` or `FALSE` to toggle all grid lines, or a string combination of `X`, `x`, `Y`, `y` for major and minor x and y grid lines.
`grid_color`	Grid line color.
`title_align, legend_align, caption_align`	Alignment of title, legend, and caption. Accepts `left`, `right`, or `center`.

Value

a ggplot theme object.

Examples

```
library(dplyr)
library(ggplot2)

fake_county |>
  filter(t_salary > 0) |>
  ggplot(aes(t_experience, t_salary)) +
  geom_point() +
  scale_y_continuous(labels = scales::dollar) +
  labs(
    title = "Salary Increases with Experience",
    subtitle = "With significant variation at all levels",
    x = "Years of Experience",
    caption = "Data from the Fake County Data Set"
  ) +
  tntp_style(family = 'sans', show_axis_titles = "x")

frpl_experience <- fake_county |>
  mutate(frpl_bucket = cut(sch_frpl_pct,
    breaks = c(0, 20, 40, 60, 80, 100),
    labels = c("0-20%", "20-40%", "40-60%", "60-80%", "80-100%")
  )) |>
  group_by(frpl_bucket) |>
  summarize(avg_experience = mean(t_experience, na.rm = TRUE)) |>
  mutate(
    label = as.character(round(avg_experience, digits = 1)),
   label = if_else(frpl_bucket == "0-20%", paste0(label, "\nYears of\nExperience"), label)
  )

frpl_experience |>
  ggplot(aes(frpl_bucket, avg_experience)) +
  geom_col(fill = if_else(frpl_experience$frpl_bucket == "60-80%",
    tntp_colors("tangerine"),
    tntp_colors("medium_gray")
  )) +
  geom_text(aes(label = label),
    nudge_y = -0.25, vjust = 1,
    color = "white", size = 5, lineheight = 1
  ) +
  labs(
    title = "High Poverty Schools have Less Experienced Teachers",
    x = "% of Student Body Receiving Free/Reduced Lunch"
  ) +
  scale_y_continuous(breaks = seq(0, 20, 4)) +
  tntp_style(
    family = 'sans',
    base_size = 20,
    show_axis_titles = "x"
  )
```

update_geom_font_defaults *Update matching font defaults for text geoms*

Description

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

Usage

```
update_geom_font_defaults(
  family = "Segoe UI",
  face = "plain",
  size = 3.5,
  color = "#2b2b2b"
)
```

Arguments

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

Value

nothing

Examples

```
# Update text geoms to use Arial font
update_geom_font_defaults(family = 'Arial')
```

update_tntpr *Re-install the tntpr package from GitHub.*

Description

Re-install the tntpr package from GitHub.

Usage

```
update_tntpr()
```

Value

nothing

Examples

```
# Run without loading tntpr first
tntpr::update_tntpr()
```

`wisc` *Fake student data from the Wisconsin State Dept. of Ed*

Description

A generated data set containing data on 1200 imaginary individual K-12 students in Wisconsin. They are nested within 6 schools in 3 districts. In adapting this from the source, Sam switched the school and district variables (there had been multiple districts per school) and made other minor changes, including dropping columns that I didn't understand or that didn't seem relevant (e.g., variables like "luck" that were used to calculate the reading and math scores).

Usage

```
wisc
```

Format

A data frame with 2700 rows and 26 variables:

student_id numeric: student's unique ID #

grade numeric: grade level

district numeric: district code

school numeric: school code

white numeric: is the student white?

black numeric: is the student black?

hisp numeric: is the student Hispanic?

indian numeric: is the student Native-American Indian?

asian numeric: is the student Asian?

econ numeric: is the student economically-disadvantaged?

female numeric: is the student female?

ell numeric: is the student an English Language Learner?

disab numeric: does the student have a learning disability?

year numeric: school year

attday numeric: days attended

readSS numeric: student's reading standardized test score

mathSS numeric: student's math standardized test score

proflvl factor: student's proficiency level

race factor: student's single-category race ...

Source

https://github.com/jknowles/r_tutorial_ed/, posted under a Creative Commons license. The script used to generate the data set is here, although not very well documented: https://github.com/jknowles/r_tutorial_ed/blob/master/data/simulate_data.R

Index