Package: bstfun (via r-universe)

October 31, 2024

Title Functions for the MSK Biostatistics Department

Version 0.5.1

Description A miscellaneous collection of functions to used by members of the Biostatistics Department at MSKCC.

License MIT + file LICENSE

URL https://github.com/MSKCC-Epi-Bio/bstfun

Depends R (>= 3.4)

Imports broom.helpers (>= 1.9.0), broom (>= 1.0.1), cli (>= 3.0.1), dplyr (>= 1.0.1), forcats (>= 0.5.0), fs (>= 1.4.2), glue (>= 1.4.1), gt (>= 0.7.0), gtsummary (>= 1.6.2), here (>= 1.0.1), lifecycle (>= 0.2.0), lubridate (>= 1.7.9), purrr (>= 0.3.4), readr (>= 1.3.1), rstudio.prefs (>= 0.1.9), rlang (>= 1.0.2), starter (>= 0.1.12), stringr (>= 1.4.0), tibble (>= 3.0.3), tidyr (>= 1.1.1)

Suggests binom, cmprsk (>= 2.2.10), covr (>= 3.5.0), flextable (>= 0.5.10), forestplot (>= 2.0.1), ggplot2 (>= 3.3.5), ggtext (>= 0.1.1), gtExtras (>= 0.4.0), Hmisc, huxtable (>= 5.4.0), kableExtra (>= 1.3.4), knitr (>= 1.33), labelled (>= 2.8.0), magick (>= 2.7.2), patchwork (>= 1.1.1), pracma, readxl (>= 1.3.1), renv (>= 0.15.1), spelling (>= 2.1), survey (>= 4.1.1), survival, testthat (>= 2.3.2), tidyverse (>= 1.3.1), usethis (>= 1.6.1)

RdMacros lifecycle

Encoding UTF-8

Language en-US

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.2

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

RemoteUrl https://github.com/ddsjoberg/bstfun

RemoteRef v0.5.1

RemoteSha 826b39f4dc2aedd000001c27fa6c3cade12f3618

Contents

`add_cuminc_risktable` *Add risk table to* `cuminc()` *plot*

Description

Plot cumulative incidence estimates with a risk table and estimates below the figure.

Usage

```
add_cuminc_risktable(
  cuminc,
  survfit,
  timepts,
  lg,
  numgrps,
  line = 3,
  at = -1,
  col.list = 1
)
```

Arguments

cuminc	cmprsk::cuminc() object
survfit	survival::survfit() object
timepts	a numeric vector of time points of the estimates to display. E.g. c(0,1,2,3,4,5) or seq(0,12,by=2)
lg	legend label for each cumulative incidence curve to be displayed. E.g. c("Male", "Female")
numgrps	the number of groups of the stratification variable: 1 is no stratification, can stratify up to 3 groups
line	to adjust position of risk table. A lower value will shift table up, a larger value will shift table down; default is 3
at	to adjust position of left margin. Default is -1.
col.list	list of colors for legend text Should match the colors of plot legend. Default is 1 (black).

Author(s)

Meier Hsu

Examples

```
library(cmprsk)
library(survival)

data(pbc)
# recode time
pbc$time.y <- pbc$time / 365.25
# recode status- switch competing events
pbc$status2 <- ifelse(pbc$status > 0, ifelse(pbc$status == 1, 2, 1), 0)
# recode stage for 3 groups
pbc$stage.3g <- ifelse(pbc$stage %in% c(1,2), "1-2", as.character(pbc$stage))


# Example 1 -----------------------------------
# CIR and KM for no strata
```

```
cif1 <- cuminc(ftime = pbc$time.y, fstatus = pbc$status2)
km1 <- survfit(Surv(pbc$time.y, pbc$status2 == 1) ~ 1)

# Plot and add risk table for no strata (numgrps=1)
windows(5, 5)
par(mfrow = c(1, 1),
    mar = c(12.5, 5.7, 2, 2),
    mgp = c(2, 0.65, 0))
plot(cif1,
     curvlab = c("recurred", "died"),
     xlim = c(0, 12), xaxt = "n")
axis(1, at = seq(0, 12, 3))

add_cuminc_risktable(cif1, km1,
                     timepts = seq(0, 12, 3),
                     lg = "",
                     numgrps = 1)

# Example 2 ------------------------------------
cif2 <- cuminc(ftime = pbc$time.y,
               fstatus = pbc$status2,
               group = pbc$sex)
km2 <- survfit(Surv(pbc$time.y, pbc$status2 == 1) ~ pbc$sex)

# Plot and add risk table for 2 groups (numgrps=2)
windows(5, 5)
par(mfrow = c(1, 1),
    mar = c(12.5, 5.7, 2, 2),
    mgp = c(2, 0.65, 0))

plot(cif2,
     curvlab = c("male", "female", "", ""),
     lty = c(1, 2, 0, 0),
     xlim = c(0, 12),
     xaxt = "n", col = c(1, 2, 0, 0))
axis(1, at = seq(0, 12, 3))

add_cuminc_risktable(cif2, km2,
                     timepts = seq(0, 12, 3),
                     lg = c("male", "female"),
                     numgrps = 2, col.list = c(1,2))

# Example 3 ------------------------------------
cif3 <- cuminc(ftime = pbc$time.y,
               fstatus = pbc$status2,
               group = pbc$stage.3g)
km3 <- survfit(Surv(pbc$time.y, pbc$status2 == 1) ~ pbc$stage.3g)
windows(6,6)
par(mfrow = c(1, 1),
    mar = c(14, 5.7, 2, 2), # change bottom margin
    mgp = c(2, 0.65, 0))

plot(cif3,
```

```
      curvlab = c("1-2", "3", "4", rep("",3)),
      lty = c(1, 2, 3, rep(0, 3)),
      xlim = c(0, 12),
      xaxt = "n", col = c(1, 2, 4, rep(0, 3)))
  axis(1, at = seq(0, 12, 3))

  add_cuminc_risktable(cif3, survfit = km3,
                       timepts = seq(0, 12, 3),
                       lg = c("1-2", "3", "4"),
                       numgrps = 3, col.list = c(1,2,4))
```

`add_inline_forest_plot` *Add inline forest plot*

Description

This function works with HTML output from the gt package only. Adds an in-line forest plot to a summary table.

Usage

```
add_inline_forest_plot(
  x,
  header = "**Forest Plot**",
  spec_pointrange.args = NULL
)
```

Arguments

`x`	a gtsummary object
`header`	string indicating column header of new forest plot column. Default is `"**Forest Plot**"`.
`spec_pointrange.args`	named list of arguments that will be passed to `kableExtra::spec_pointrange()`. Use this argument to modify the default ascetics of the forest plot, e.g. color, size, symbols, etc. Default is `list(width = 250, cex = .75, col = "black", pch = 16)`

Details

Estimates from `tbl_regression()` *and* `tbl_uvregression()` *that have been exponentiated are shown on the log scale.*

Value

gtsummary object

Example Output

See Also

Other gtsummary-related functions: add_sparkline(), as_ggplot(), bold_italicize_group_labels(), logistic_reg_adj_diff(), style_tbl_compact(), tbl_likert(), theme_gtsummary_msk()

Examples

```
library(gtsummary)

# Example 1 ----------------------------------
add_inline_forest_plot_ex1 <-
  lm(mpg ~ cyl + am + drat, mtcars) %>%
  tbl_regression() %>%
  add_inline_forest_plot()
```

---

add_sparkline *Add Sparkline Figure*

Description

This function wraps gtExtras::gt_plt_dist() and adds a new column illustrating the distribution of a continuous variable. This function converts the gtsummary table into a gt table.

Usage

```
add_sparkline(
  x,
  type = c("boxplot", "histogram", "rug_strip", "density", "sparkline"),
  column_header = NULL,
  same_limit = FALSE,
  ...
)
```

Arguments

x	'tbl_summary' object
type	sparkline type. Must be one of c("boxplot", "histogram", "rug_strip", "density", "sparkline")
column_header	string indicating column header
same_limit	A logical indicating that the plots will use the same axis range (TRUE) or have individual axis ranges (FALSE).
...	Arguments passed on to gtExtras::gt_plt_dist

`fig_dim` A vector of two numbers indicating the height/width of the plot in mm at a DPI of 25.4, defaults to `c(5,30)`

`line_color` Color for the line, defaults to `"black"`. Accepts a named color (eg 'blue') or a hex color.

`fill_color` Color for the fill of histograms/density plots, defaults to `"grey"`. Accepts a named color (eg `'blue'`) or a hex color.

`bw` The bandwidth or binwidth, passed to `density()` or `ggplot2::geom_histogram()`. If `type = "density"`, then `bw` is passed to the `bw` argument, if `type = "histogram"`, then `bw` is passed to the `binwidth` argument.

`trim` A logical indicating whether to trim the values in `type = "density"` to a slight expansion beyond the observable range. Can help with long tails in `density` plots.

Value

a gt table

Example Output

See Also

Other gtsummary-related functions: `add_inline_forest_plot()`, `as_ggplot()`, `bold_italicize_group_labels()`, `logistic_reg_adj_diff()`, `style_tbl_compact()`, `tbl_likert()`, `theme_gtsummary_msk()`

Examples

```
library(gtsummary)

add_sparkline_ex1 <-
  trial %>%
  select(age, marker) %>%
  tbl_summary(missing = "no") %>%
  add_sparkline()
```

---

add_splines *Add spline terms to a data frame*

Description

Adds spline terms calculated via `Hmisc::rcspline.eval()` to a data frame.

Usage

```
add_splines(data, variable, knots = NULL, nk = 5, norm = 2, new_names = NULL)
```

Arguments

data	a data frame
variable	name of column in data
knots	knot locations. If not given, knots will be estimated using default quantiles of x. For 3 knots, the outer quantiles used are 0.10 and 0.90. For 4-6 knots, the outer quantiles used are 0.05 and 0.95. For nk $>$ 6, the outer quantiles are 0.025 and 0.975. The knots are equally spaced between these on the quantile scale. For fewer than 100 non-missing values of x, the outer knots are the 5th smallest and largest x.
nk	number of knots. Default is 5. The minimum value is 3.
norm	'0' to use the terms as originally given by *Devlin and Weeks (1986)*, '1' to normalize non-linear terms by the cube of the spacing between the last two knots, '2' to normalize by the square of the spacing between the first and last knots (the default). norm=2 has the advantage of making all nonlinear terms beon the x-scale.
new_names	Optionally specify names of new spline columns

Value

data frame

Knot Locations

Knot locations are returned in attr(data[[variable]], "knots")

Examples

```
trial %>%
  add_splines(age)
```

---

add_variable_grouping *Group variable summaries*

Description

Some data are inherently grouped, and should be reported together. For example, one person likely belongs to multiple racial groups and the results of these tabulations belong in a grouped portion of a summary table.

Grouped variables are all indented together. The label row is a single indent, and the other rows are double indented.

Usage

```
add_variable_grouping(x, ...)
```

Arguments

x	a gtsummary table
...	named arguments. The name is the group label that will be inserted into the table. The values are character names of variables that will be grouped

Value

a gtsummary table

Warning

While the returned table is the same class as the input, it does not follow the structure expected in other gtsummary functions that accept these objects: errors may occur.

Example Output

Examples

```
set.seed(11234)
add_variable_grouping_ex1 <-
  data.frame(
    race_asian = sample(c(TRUE, FALSE), 20, replace = TRUE),
    race_black = sample(c(TRUE, FALSE), 20, replace = TRUE),
    race_white = sample(c(TRUE, FALSE), 20, replace = TRUE),
    age = rnorm(20, mean = 50, sd = 10)
  ) %>%
  gtsummary::tbl_summary(
    label = list(race_asian = "Asian",
                 race_black = "Black",
                 race_white = "White",
                 age = "Age")
  ) %>%
  add_variable_grouping(
    "Race (check all that apply)" = c("race_asian", "race_black", "race_white")
  )
```

---

assign_timepoint *Assign a time point to a long data set with multiple measures*

Description

Given a data set that has a measure collected over time and you want to extract, for example the 3 month measurement, this function will find the measure closest to 3 months within a defined window.

Usage

```
assign_timepoint(
  data,
  id,
  ref_date,
  measure_date,
  timepoints,
  windows,
  time_units = c("days", "weeks", "months", "years"),
  new_var = "timepoint",
  keep_all_obs = FALSE,
  keep_all_vars = TRUE
)
```

Arguments

data	data frame
id	id variable name, such as "mrn"
ref_date	baseline or reference date column name
measure_date	date the measure was collected
timepoints	vector of time point to identify
windows	list of windows around a time point that are acceptable
time_units	one of c("days", "weeks", "months", "years")
new_var	name of new variable, default is "timepoint"
keep_all_obs	logical indicating whether to return a data frame with only the assigned time points (default), or to return a data frame with all rows.
keep_all_vars	logical indicating whether to return a data frame with all the variables in data= and the new time point column (default), or a limited data frame including only the column involved in assigning a time point.

Value

data frame passed in data with additional column new_var

Examples

```
ggplot2::economics_long %>%
  dplyr::group_by(variable) %>%
  dplyr::mutate(min_date = min(date)) %>%
  dplyr::ungroup() %>%
  assign_timepoint(
    id = variable,
    ref_date = min_date,
    measure_date = date,
    timepoints = c(6, 12, 24),
    windows = list(c(-2, 2), c(-2, 2), c(-2, 2)),
    time_units = "months"
  )
```

`as_forest_plot` *Create Forest Plot*

Description

[Experimental] This function takes a gtsummary table and converts it to a forest plot using `forestplot::forestplot()`.

Usage

```
as_forest_plot(
  x,
  col_names = c("estimate", "ci", "p.value"),
  graph.pos = 2,
  boxsize = 0.3,
  title_line_color = "darkblue",
  xlog = x$inputs$exponentiate,
  ...
)
```

Arguments

`x`	a gtsummary object of class `"tbl_regression"` or `"tbl_uvregression"`
`col_names`	names of columns in `x$table_body` to print on the RHS of the forest plot. Default is `c("estimate", "ci", "p.value")`
`graph.pos`	The position of the graph element within the table of text. The position can be `1-(ncol(labeltext) + 1)`. You can also choose set the position to `"left"` or `"right"`.
`boxsize`	Override the default box size based on precision
`title_line_color`	color of line that appears above forest plot. Default is `"darkblue"`
`xlog`	If TRUE, x-axis tick marks are to follow a logarithmic scale, e.g. for logistic regression (OR), survival estimates (HR), Poisson regression etc. *Note:* This is an intentional break with the original `forestplot` function as I've found that exponentiated ticks/clips/zero effect are more difficult to for non-statisticians and there are sometimes issues with rounding the tick marks properly.
`...`	arguments passed to `forestplot::forestplot()`

Author(s)

Christine Zhou

Examples

```
library(gtsummary)
library(survival)

# Example 1 ----------------------------------
tbl_uvregression(
  trial[c("response", "age", "grade")],
  method = glm,
  y = response,
  method.args = list(family = binomial),
  exponentiate = TRUE
) %>%
as_forest_plot()

# Example 2 ----------------------------------
tbl <-
 coxph(Surv(ttdeath, death) ~ age + marker, trial) %>%
 tbl_regression(exponentiate = TRUE) %>%
 add_n()

as_forest_plot(tbl, col_names = c("stat_n", "estimate", "ci", "p.value"))

# Example 3 ----------------------------------
tbl %>%
modify_cols_merge(
  pattern = "{estimate} ({ci})",
  rows = !is.na(estimate)
) %>%
modify_header(estimate = "HR (95% CI)") %>%
as_forest_plot(
  col_names = c("estimate", "p.value"),
  boxsize = 0.2,
  col = forestplot::fpColors(box = "darkred")
)
```

---

as_ggplot *Convert gt/gtsummary table to ggplot*

Description

useful when you want to place a ggplot and gt table side-by-side. To use this function you must install the magick R package AND system program (see https://docs.ropensci.org/magick/articles/intro.html#installing-magick-1)

Usage

```
as_ggplot(x, ...)
```

Arguments

x	gt or gtsummary table
...	arguments passed to `gt::gtsave()`

Value

a ggplot object

See Also

Other gtsummary-related functions: `add_inline_forest_plot()`, `add_sparkline()`, `bold_italicize_group_labels()`, `logistic_reg_adj_diff()`, `style_tbl_compact()`, `tbl_likert()`, `theme_gtsummary_msk()`

Examples

```
library(gtsummary)
library(ggplot2)
library(patchwork)

# # convert gtsummary table to ggplot
# tbl <-
#  trial %>%
#  select(age, marker, trt) %>%
#  tbl_summary(by = trt, missing = "no") %>%
#  as_ggplot()
#
# # create basic ggplot
# gg <-
#  trial %>%
#  ggplot(aes(x = age, y = marker, color = trt)) +
#  geom_point()
#
# # stack tables using patchwork
# gg / tbl
```

---

auc *Calculate exact AUCs based on the distribution of risk in a population*

Description

Provided a distribution of risk in a population, this function calculates the exact AUC of a model that produces the risk estimates. For example, a logistic regression model built with a normal linear predictor yields logit-normal distributed predicted risks. The AUC from the logistic regression model is the same as the AUC estimated from the distribution of the predicted risks, independent of the outcome. This method for AUC calculation is useful for simulation studies where the predicted risks are a mixture of two distributions. The exact prevalence of the outcome can easily be calculated, along with the exact AUC of the model.

Usage

```
auc_density(density, cut.points = seq(from = 0, to = 1, by = 0.001), ...)

auc_histogram(x)
```

Arguments

density	a function name that describes the continuous probability density function of the risk from 0 to 1.
cut.points	sequence of points in [0, 1] where the sensitivity and specificity are calculated. More points lead to a more precise estimate of the AUC. Default is seq(from = 0, to = 1,by = 0.001).
...	arguments for the function specified in density. For example, dbeta(x, shape1=1, shape2=1) has need for two additional arguments to specify the density function (shape1 and shape2).
x	histogram object from graphics::hist

Value

Returns a list sensitivity and specificity at each cut point, the expected value or mean risk, and the AUC associated with the distribution.

Author(s)

Daniel D Sjoberg

Examples

```
auc_density(density = dbeta, shape1 = 1, shape2 = 1)

runif(10000) %>%
  hist(breaks = 250) %>%
  auc_histogram()
```

---

bold_italicize_group_labels

Set bold and/or italic style for groups labels in stacked tables

---

Description

Set bold and/or italic style for groups labels in stacked tables

Usage

```
bold_italicize_group_labels(
  x,
  bold = FALSE,
  italics = FALSE,
  print_engine = c("gt", "flextable", "huxtable")
)
```

Arguments

x	a gtsummary stacked table
bold	logical indicating whether to bold the group header rows
italics	logical indicating whether to italicize the group header rows
print_engine	Choose a print engine to render the table, must be one of c("gt", "flextable", "huxtable")

Value

A table of class selected in print_engine. Of note, the output will no longer be a gtsummary table.

Example Output

See Also

Other gtsummary-related functions: add_inline_forest_plot(), add_sparkline(), as_ggplot(), logistic_reg_adj_diff(), style_tbl_compact(), tbl_likert(), theme_gtsummary_msk()

Examples

```
library(gtsummary)

bold_italicize_group_labels_ex1 <-
  trial %>%
  select(age, trt, grade) %>%
  tbl_strata(
    strata = grade,
    ~ .x %>%
      tbl_summary(by = trt, missing = "no"),
    .combine_with = "tbl_stack"
  ) %>%
  bold_italicize_group_labels(bold = TRUE)
```

`cite_r` *Cite R and R Packages*

Description

Add citations to R and R packages in an R Markdown report.

Usage

```
cite_r(pkgs = c("tidyverse", "gtsummary"), add_citations = TRUE)
```

Arguments

`pkgs`	character vector of package names to cite. Default is `c("tidyverse", "gtsummary")`. `NULL` is acceptable.
`add_citations`	logical indicating whether to include the bibtex citations of R and the packages. Default is `TRUE`. When TRUE, we expect the R markdown file to reference a bib file including the references for each package listed in the `pkgs=` argument. The cite key for each of these entries must match the package name. Also, the file must include an entry for R with cite key 'r'.

R Markdown

Below is an example how the `cite_r()` function would be used in an R Markdown report.

```
---
output: html_document
bibliography: references.bib
---

Analyses were conducted with `r bstfun::cite_r(pkgs = "tidyverse")`.
```

This assumes there is a bib file of references called `references.bib` that contain the following entries.

```
@Manual{r,
  title = {R: A Language and Environment for Statistical Computing},
  author = {{R Core Team}},
  organization = {R Foundation for Statistical Computing},
  address = {Vienna, Austria},
  year = {2021},
  url = {https://www.R-project.org/},
}

@Article{tidyverse,
  title = {Welcome to the {tidyverse}},
```

```
 author = {Hadley Wickham and Mara Averick and Jennifer Bryan and Winston Chang and Lucy D'Agostino McG
  year = {2019},
  journal = {Journal of Open Source Software},
  volume = {4},
  number = {43},
  pages = {1686},
  doi = {10.21105/joss.01686},
}
```

Examples

```
# cite R and the tidyverse
cite_r(pkgs = "tidyverse")

# cite R and the tidyverse, but text only
cite_r(pkgs = "tidyverse", add_citations = FALSE)

# only cite R
cite_r(pkgs = NULL)
```

---

clean_mrn *Check and Format MRNs*

Description

An MRN follows specific rules

1. Must be character
2. Must contain only numeric components
3. Must be eight characters long and include leading zeros.

This function converts numeric MRNs to character and ensures it follows MRN conventions. Character MRNs can also be passed, and leading zeros will be appended and checked for consistency.

Usage

```
clean_mrn(x, allow_na = FALSE, check_unique = FALSE)
```

Arguments

x	vector to be converted and checked to MRN
allow_na	logical indicating whether NA values are accepted. Default is FALSE
check_unique	Check if MRNs are unique

Value

character MRN vector

Examples

```
1000:1001 %>%
  clean_mrn()
```

`count_map` *Check variable derivations*

Description

Function assists in checking the values of new, derived variables against the raw, source variables.

Usage

```
count_map(data, ...)
```

Arguments

`data`	data frame
`...`	sets of variables to check. variables that are checked together are included in the same vector. See example below.

Examples

```
count_map(
  mtcars,
  c(cyl, am), c(gear, carb)
)
```

`count_na` *Assess pattern of missing data*

Description

Pass a data frame and the missing pattern of all columns in the data frame. The data frame is returned unmodified.

Usage

```
count_na(data, include = NULL, exclude = NULL)
```

Arguments

`data`	data frame
`include`	character vector of names to include
`exclude`	character vector of names to exclude

Value

original data frame invisibly returned

Examples

```
trial %>% count_na()
```

`create_project` *Start a New Biostatistics project*

Description

Creates a directory with the essential files for a new project. The function can be used on existing project directories as well.

The folder name should be structured as "<PI Last Name> <Short Description>", e.g. "Sjoberg MRI detects Path Stage after Surgery". PI last name is used in file naming.

Usage

```
create_bst_project(path, path_data = NULL, git = NA, ...)
```

Arguments

path	A path. If it exists, it is used. If it does not exist, it is created.
path_data	A path. The directory where the secure data exist. Default is NULL. When supplied, a symbolic link to data folder will be created.
git	Logical indicating whether to create Git repository. Default is TRUE When NA, user will be prompted whether to initialise Git repo.
...	Arguments passed on to starter::create_project renv Logical indicating whether to add renv to a project. Default is TRUE. When NA user is asked interactively for preference. symlink Logical indicating whether to place a symbolic link to the location in path_data=. Default is to place the symbolic link if the project is a git repository. renv.settings A list of renv settings passed to renv::scaffold(settings=) overwrite Logical indicating whether to overwrite existing files if they exist. Options are TRUE, FALSE, and NA (aka ask interactively). Default is NA open Logical indicating whether to open new project in fresh RStudio session

See Also

starter::create_project()

Examples

```
# specifying project folder location (folder does not yet exist)
project_path <- fs::path(tempdir(), "My Project Folder")

# creating folder where secure data would be stored (typically will be a network drive)
secure_data_path <- fs::path(tempdir(), "secure_data")
dir.create(secure_data_path)

# creating new project folder
create_bst_project(project_path, path_data = secure_data_path)
```

---

egfr *Calculate eGFR*

Description

Calculate eGFR

Usage

```
egfr_ckdepi(creatinine, age, female, aa, label = "eGFR, mL/min/1.73m²")

egfr_mdrd(creatinine, age, female, aa, label = "eGFR, mL/min/1.73m²")
```

Arguments

creatinine	serum creatinine level in mg/dL
age	patient age
female	logical indicating whether patient is female
aa	logical indicating whether patient is African-American
label	label that will be applied to result, e.g. `attr("label", 'eGFR, mL/min/1.73m\U00B2')`

Value

numeric vector

Examples

```
egfr_mdrd(creatinine = 1.2, age = 60, female = TRUE, aa = TRUE)
egfr_ckdepi(creatinine = 1.2, age = 60, female = TRUE, aa = TRUE)
```

`fix_database_error` *Function to make database fixes after import*

Description

Use this function to make updates to your data while avoiding adding PHI, such as MRNs, to your code and subsequently PHI in GitHub. You provide a file of database fixes that is three columns: 1. An expression that selects a line in the database to update (e.g. `MRN == "12345678"`), 2. The column name that will be updated, and 3. The updated value. See the examples for the structure of the database fix input.

Usage

```
fix_database_error(data, engine = readr::read_csv, ...)
```

Arguments

`data`	data frame with errors
`engine`	function to import file of database fixes
`...`	arguments passed to the engine function to import the database fixes

Value

updated data frame

Examples

```
df_fixes <-
  tibble::tribble(
    ~id, ~variable, ~value,
    "id == 1", "age", "56",
    "id == 2", "trt", "Drug C"
  )
trial %>%
  dplyr::mutate(id = dplyr::row_number()) %>%
  fix_database_error(
    engine = I,
    x = df_fixes
  )
```

`followup_time` *Report Follow-up Among Censored Obs*

Description

Function accepts `survival::Surv()` object, extracts the follow-up times among the censored observations, and returns the requested summary statistics. Use this function to report follow-up in-line in R Markdown reports.

Usage

```
followup_time(
  Surv,
  data = NULL,
  pattern = "{median} (IQR {p25}, {p75})",
  style_fun = gtsummary::style_sigfig
)
```

Arguments

`Surv`	An object of class `"Surv"` created with `survival::Surv()`—not a multi-state endpoint.
`data`	A data frame
`pattern`	Statistics pattern to return. Default is `"{median} (IQR {p25}, {p75})"`. User may select the following summary statistics to report, `"{median}"`, `"{mean}"`, `"{p25}"`, `"{p75}"`, `"{sd}"`, `"{var}"`, `"{min}"`, and `"{max}"`. Also available are `"{n}"` (the number of events) and `"{N}"` (number of non-missing observations).
`style_fun`	Function used to style/format the summary statistics. Default is `gtsummary::style_sigfig`. Argument accepts anonymous function notation, e.g. `~gtsummary::style_sigfig(., digits = 3)`

Value

string of summary statistics

Examples

```
library(survival)

followup_time(Surv(time, status), data = lung)

followup_time(
  Surv(time, status), data = lung,
  pattern = "{median} days",
  style_fun = ~gtsummary::style_sigfig(., digits = 4)
)
```

```
followup_time(
  survival::Surv(time, status),
  data = survival::lung,
  pattern = "{n} events with a follow-up time of {median} days (IQR {p25}, {p75})"
)
```

`get_mode` *Calculates the mode(s) of a set of values*

Description

This function calculates the most common value(s) of a given set

Usage

```
get_mode(x, moden = 1, quiet = FALSE)
```

Arguments

`x`	A variable or vector (numeric, character or factor)
`moden`	If there are multiple modes, which mode to use. The default is the first mode.
`quiet`	By default, messages are printed if multiple modes are selected. To hide these messages, set `quiet` to `TRUE`

Value

A vector of length 1 containing the mode

Examples

```
get_mode(trial$stage)
get_mode(trial$trt)
get_mode(trial$response)
get_mode(trial$grade)
```

`ggcalibration` *Model Calibration Plot*

Description

Assess a model's calibration via a calibration plot.

Usage

```
ggcalibration(
  data,
  y,
  x,
  n.groups = 10,
  conf.level = 0.95,
 ci.method = c("exact", "ac", "asymptotic", "wilson", "prop.test", "bayes", "logit",
    "cloglog", "probit"),
 geom_smooth.args = list(method = "loess", se = FALSE, formula = y ~ x, color = "black"),
  geom_errorbar.args = list(width = 0),
  geom_point.args = list(),
  geom_function.args = list(colour = "gray", linetype = "dashed")
)
```

Arguments

`data`	a data frame
`y`	variable name of the outcome coded as 0/1
`x`	variable name of the risk predictions
`n.groups`	number of groups
`conf.level`	level of confidence to be used in the confidence interval
`ci.method`	method to use to construct the interval. See `binom::binom.confint()` for details
`geom_smooth.args`	named list of arguments that will be passed to `ggplot2::geom_smooth()`. Default is `list(method = "loess", se = FALSE, formula = y ~ x, color = "black")`
`geom_errorbar.args`	named list of arguments that will be passed to `ggplot2::geom_errorbar()`. Default is `list(width = 0)`
`geom_point.args`	named list of arguments that will be passed to `ggplot2::geom_point()`. Default is `list()`
`geom_function.args`	named list of arguments that will be passed to `ggplot2::geom_function()` and is the function that adds the 45 degree guideline. Default is `list(colour = "gray", linetype = "dashed")`

Value

ggplot

Examples

```
glm(response ~ age + marker + grade, trial, family = binomial) %>%
  broom::augment(type.predict = "response") %>%
  ggcalibration(y = response, x = .fitted, n.groups = 6) +
  ggplot2::xlim(0, 1) +
  ggplot2::labs(x = "Model Risk")
```

---

here_data *Find your data folder*

Description

Uses `data_date.txt` to create a path with the data date populated.

- `here_data()`: Returns `here::here("secure_data", data_date, ...)`
- `path_data()`: Returns `fs::path(path, "secure_data", data_date, ...)`

Usage

```
here_data(
  ...,
  data_folder_name = "secure_data",
  path_to_data_date = here::here()
)

path_data(
  ...,
  path = getOption("path_data"),
  data_folder_name = "secure_data",
  path_to_data_date = here::here()
)

get_data_date(path_to_data_date = here::here())
```

Arguments

`...` Path components to be appended to the end of the returned path string.

`data_folder_name`
name of data folder. Default is `"secure_data"`

`path_to_data_date`
path to data date folder or file. If folder is passed, expecting the data date file to be named one of `c("data_date.txt", "data_date", "dataDate.txt", "dataDate")`. Default value is `here::here()`

path — path to folder where data is saved, e.g. `fs::path(path, )` Default value is `getOption("path_data")` where `path_data` can be initialized at the top of the script with `options(path_data = "H:/.../data")`

Details

The function expects the user to version their data using a text file indicating the date the data was last received, and the data to be stored in a corresponding folder name, e.g. `~/Project Folder/secure_data/2020-01-01`.

Value

path to data folder

Examples

```
if (FALSE) {
here_data()
#> "C:/Users/SjobergD/GitHub/My Project/secure_data/2020-01-01"

here_data("Raw Data.xlsx")
#> "C:/Users/SjobergD/GitHub/My Project/secure_data/2020-01-01/Raw Data.xlsx"

path_data(path = "O:/My Project", "Raw Data.xlsx")
#> "O:/My Project/secure_data/2020-01-01/Raw Data.xlsx"
}
```

---

hpcc *HPCC Functions*

Description

Functions for working with the high performance computing cluster

- `hpcc_get_arg()` retrieve character argument passed from terminal, e.g. `qsubR bootstrap_analysis.R "mets"`
- `hpcc_get_seq_number()` retrieve sequence integer when a sequence of jobs were submitted from the terminal, e.g `qsubR -a 1-50%10 bootstrap_analysis.R`

Usage

```
hpcc_get_arg()

hpcc_get_seq_number()
```

list_labels *Get variable labels and store in named list*

Description

Get variable labels and store in named list

Usage

```
list_labels(data)
```

Arguments

data	Data frame

Author(s)

Daniel D. Sjoberg

Examples

```
list_labels(trial)
```

logistic_reg_adj_diff *Logistic regression adjusted differences*

Description

This function works with `gtsummary::add_difference()` to calculate adjusted differences and confidence intervals based on results from a logistic regression model. Adjustment covariates are set to the mean to estimate the adjusted difference. The function uses bootstrap methods to estimate the adjusted difference between two groups. The CI is estimate by either using the SD from the bootstrap difference estimates and calculating the CI assuming normality or using the centiles of the bootstrapped differences as the confidence limits

The function can also be used in `add_p()`, and if you do, be sure to set `boot_n = 1` to avoid long, unused computation.

Usage

```
logistic_reg_adj_diff(
  data,
  variable,
  by,
  adj.vars,
  conf.level,
  type,
```

```
  ci_type = c("sd", "centile"),
  boot_n = 250,
  ...
)
```

Arguments

`data`	a data frame
`variable`	string of binary variable in `data=`
`by`	string of the `by=` variable name
`adj.vars`	character vector of variable names to adjust model for
`conf.level`	Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
`type`	string indicating the summary type
`ci_type`	string dictation bootstrap method for CI estimation. Must be one of `c("sd", "centile")`.
`boot_n`	number of bootstrap iterations to use. In most cases, it is reasonable to used 250 for the `"sd"` method and 5000 for the `"centile"` method.
`...`	not used

Value

tibble with difference estimate

Example Output

See Also

Other gtsummary-related functions: `add_inline_forest_plot()`, `add_sparkline()`, `as_ggplot()`, `bold_italicize_group_labels()`, `style_tbl_compact()`, `tbl_likert()`, `theme_gtsummary_msk()`

Examples

```
library(gtsummary)
tbl <- tbl_summary(trial, by = trt, include  = response, missing = "no")

# Example 1 ----------------------------------------------------------------
logistic_reg_adj_diff_ex1 <-
  tbl %>%
  add_difference(
    test = everything() ~ logistic_reg_adj_diff,
    adj.vars = "stage"
  )

# Example 2 ----------------------------------------------------------------
# Use the centile method, and
```

```
# change the number of bootstrap resamples to perform
logistic_reg_adj_diff_ex2 <-
  tbl %>%
  add_difference(
    test = everything() ~
      purrr::partial(logistic_reg_adj_diff, ci_type = "centile", boot_n = 100),
    adj.vars = "stage"
  )
```

`project_templates` *Biostatistics project templates*

Description

The `project_templates` object defines the contents of the Biostatistics project templates used in `create_bst_project()` and `use_bst_file()`.

Usage

```
project_templates
```

Format

A named list containing the Biostatistics project template.

See Also

`create_bst_project()`

`use_bst_file()`

Examples

```
if (FALSE) {
create_hot_project(
  path = file.path(tempdir(), "Sjoberg New Project"),
  template = bstfun::project_templates[["default"]]
)
}
```

rm_logs *Deletes log files created by Rscript on the Linux cluster*

Description

When an R script is submitted to the server, Linux generates a log file named `myRscript.R.o######`. This program searches a folder for files named like this and removes/deletes them.

Usage

```
rm_logs(path = here::here(), recursive = FALSE)
```

Arguments

`path`	folder location to search for log files
`recursive`	logical. should log files in subdirectories also be deleted?

set_derived_variables *Apply variable labels to data frame*

Description

Takes labels from the Derived Variables excel file and applies them to the passed data frame. The excel sheet must have columns `"varname"` and `"label"`.

Usage

```
set_derived_variables(data, path, sheet = NULL, drop = TRUE)
```

Arguments

`data`	Data frame
`path`	Path to Derived Variables xls/xlsx file
`sheet`	Sheet to read. Either a string (the name of a sheet), or an integer (the position of the sheet). Ignored if the sheet is specified via `range`. If neither argument specifies the sheet, defaults to the first sheet.
`drop`	Logical indicating whether to drop unlabelled variables

Author(s)

Daniel D. Sjoberg

Examples

```
trial %>%
  set_derived_variables("derived_variables_sjoberg.xlsx")
```

style_tbl_compact *Compact Table Styling*

Description

Apply the same compact styling available from `gtsummary::theme_gtsummary_compact()` to any gt, flextable, huxtable, or `knitr::kable()` table. `knitr::kable()` stlying uses the kableExtra package

Usage

```
style_tbl_compact(data)
```

Arguments

`data`	a gt, flextable, huxtable, or `knitr::kable()` table.

Example Output

See Also

Other gtsummary-related functions: `add_inline_forest_plot()`, `add_sparkline()`, `as_ggplot()`, `bold_italicize_group_labels()`, `logistic_reg_adj_diff()`, `tbl_likert()`, `theme_gtsummary_msk()`

Examples

```
style_tbl_compact_ex1 <-
  head(trial) %>%
  gt::gt() %>%
  style_tbl_compact()
```

tbl_likert *Likert Summary Table*

Description

[Experimental]

`tbl_likert()` creates a summary of Likert scales following the gtsummary structure.

`add_n()` adds a column to the table with the total number of observations.

`add_continuous_stat()` converts Likert scales into a numeric score and computes continuous statistics based on this score.

Usage

```
tbl_likert(
  data,
  label = NULL,
  statistic = NULL,
  digits = NULL,
  include = everything(),
  sort = c("default", "ascending", "descending")
)

## S3 method for class 'tbl_likert'
add_n(
  x,
  statistic = "{n}",
  col_label = "**N**",
  footnote = FALSE,
  last = FALSE,
  ...
)

add_continuous_stat(x, ...)

## S3 method for class 'tbl_likert'
add_continuous_stat(
  x,
  statistic = "{mean}",
  digits = NULL,
  col_label = NULL,
  footnote = FALSE,
  last = TRUE,
  score_values = NULL,
  stat_col_name = NULL,
  ...
)
```

Arguments

`data`	A data frame
`label`	List of formulas specifying variables labels, e.g. `list(age ~ "Age", stage ~ "Path T Stage")`. If a variable's label is not specified here, the label attribute (`attr(data$age, "label")`) is used. If attribute label is `NULL`, the variable name will be used.
`statistic`	String or formula indicating the statistic to be reported. Default is the mean score. Other possible continuous statistics are described in `gtsummary::tbl_summary()` help page, section *statistic argument*.
`digits`	Formula or list of formulas indicating how to display the computed statistics, see `gtsummary::tbl_summary()` help page

include	variables to include in the summary table. Default is everything()
sort	Sort table based on mean scores? Must be one of c("default", "ascending", "descending")
x	Object with class tbl_likert from the tbl_likert() function
col_label	String indicating the column label. Default is generated from statistic.
footnote	Logical argument indicating whether to print a footnote clarifying the statistics presented. Default is FALSE
last	Logical indicator to include the new column last in table. Default is TRUE
...	not used
score_values	Vector indicating the numeric value of each factor level. Default is 1:n where n indicates the number of levels.
stat_col_name	Optional string indicating the name of the new column added to x$table_body

Example Output

See Also

Other gtsummary-related functions: add_inline_forest_plot(), add_sparkline(), as_ggplot(), bold_italicize_group_labels(), logistic_reg_adj_diff(), style_tbl_compact(), theme_gtsummary_msk()

Examples

```
library(dplyr)
set.seed(1123)
likert_lvls <- c("Never", "Sometimes", "Often", "Always")

df <-
  tibble::tibble(
    Q1 = sample(likert_lvls, size = 100, replace = TRUE),
    Q2 = sample(likert_lvls, size = 100, replace = TRUE)
  ) %>%
  mutate(across(c(Q1, Q2), ~factor(., levels = likert_lvls)))

tbl_likert_ex1 <-
  tbl_likert(df) %>%
  add_n() %>%
  add_continuous_stat(statistic = "{mean}") %>%
  add_continuous_stat(statistic = "{sd}")
```

theme_gtsummary_msk *Set custom gtsummary themes*

Description

This is a place for any member of the MSK community to add a personal gtsummary theme. Reach out if you're interested in adding yours!

Usage

```
theme_gtsummary_msk(
 name = c("hot", "karissa", "ally", "mauguen", "esther", "curry", "lavery", "meier",
  "leej", "whitingk", "kwhiting", "mauguena", "drille", "currym1", "laveryj", "hsum1",
   "leej22"),
  font_size = NULL
)
```

Arguments

name	string indicating the custom theme to set.
font_size	Numeric font size for compact theme. Default is 13 for gt tables, and 8 for all other output types

Details

Visit the gtsummary themes vignette for a full list of preferences that can be set.

See Also

Other gtsummary-related functions: add_inline_forest_plot(), add_sparkline(), as_ggplot(), bold_italicize_group_labels(), logistic_reg_adj_diff(), style_tbl_compact(), tbl_likert()

Examples

```
theme_gtsummary_msk("hot")
```

trial *Results from a simulated study of two chemotherapy agents: Drug A and Drug B*

Description

A dataset containing the baseline characteristics of 200 patients who received Drug A or Drug B. Dataset also contains the outcome of tumor response to the treatment.

Usage

```
trial
```

Format

A data frame with 200 rows–one row per patient

trt Chemotherapy Treatment

age Age, yrs

marker Marker Level, ng/mL

stage T Stage

grade Grade

response Tumor Response

death Patient Died

ttdeath Months to Death/Censor

---

use_bst_rstudio_prefs *Use Biostatistics RStudio Preferences*

Description

The function wraps `rstudio.prefs::use_rstudio_prefs()` and sets the following preferences in RStudio.

Preference	Value
always_save_history	FALSE
graphics_backend	ragg
load_workspace	FALSE
show_hidden_files	TRUE
show_line_numbers	TRUE
margin_column	80
save_workspace	never
rainbow_parentheses	TRUE
restore_last_project	FALSE
rmd_chunk_output_inline	FALSE
show_last_dot_value	TRUE
show_margin	TRUE
show_invisibles	TRUE

Usage

```
use_bst_rstudio_prefs(profile = tolower(Sys.info()[["user"]]))
```

Arguments

profile	Used to set additional preferences saved under specified profile. Default is your current system username. If no profile exists, it is ignored.

Examples

```
use_bst_rstudio_prefs()
```

---

use_file *Write a template file*

Description

Rather than using `create_bst_project()` to start a new project folder, you may use `use_bst_file()` to write a single file from any project template. The functions `use_bst_gitignore()` and `use_bst_readme()` are shortcuts for `use_bst_file("gitignore")` and `use_bst_file("readme")`.

Usage

```
use_bst_file(name = NULL, filename = NULL, open = interactive())

use_bst_gitignore(filename = NULL)

use_bst_readme(filename = NULL)

use_bst_setup(filename = NULL)

use_bst_analysis(filename = NULL)

use_bst_report(filename = NULL)
```

Arguments

name	Name of file to write. Not sure of the files available to you? Run the function without specifying a name, and all files available within the template will print.
filename	Optional argument to specify the name of the file to be written. Paths/filename is relative to project base (e.g. `here::here()`)
open	If `TRUE`, opens the new file.

See Also

`create_bst_project()`

Examples

```
if (FALSE) {
# create gitignore file
use_bst_file("gitignore")
use_bst_gitignore()

# create README.md file
use_bst_file("readme")
use_bst_readme()
}
```

`use_varnames_as_labels` *Use Variable Names as Labels*

Description

Use Variable Names as Labels

Usage

```
use_varnames_as_labels(data, caps = NULL, exclude = NULL)
```

Arguments

`data`	a data frame
`caps`	variables to be entirely capitalized
`exclude`	variables to exclude from labeling

Value

a labeled data frame

Examples

```
library(gtsummary)

mtcars %>%
  select(mpg, cyl, vs, am, carb) %>%
  use_varnames_as_labels(caps = c(mpg, vs, am), exclude = cyl) %>%
  tbl_summary() %>%
  as_kable(format = "simple")
```

Index