Package 'ggsurvfit'

October 16, 2022

Title Flexible Time-to-Event Figures

Version 0.2.0

Description Ease the creation of time-to-event (i.e. survival) endpoint figures. The modular functions create figures ready for publication. Each of the functions that add to or modify the figure are written as proper 'ggplot2' geoms or stat methods, allowing the functions from this package to be combined with any function or customization from 'ggplot2' and other 'ggplot2' extension packages.

License MIT + file LICENSE

URL https://github.com/ddsjoberg/ggsurvfit, http://www.danieldsjoberg.com/ggsurvfit/

BugReports https://github.com/ddsjoberg/ggsurvfit/issues

Depends R (>= 3.5)

Imports broom (>= 1.0.0), cli (>= 3.0.0), dplyr (>= 1.0.3), ggplot2 (>= 3.3.0), glue (>= 1.6.0), gtable, patchwork (>= 1.1.0), rlang (>= 1.0.0), scales (>= 1.2.0), survival (>= 3.4-0), tidyr (>= 1.0.0)

Suggests covr, knitr, rmarkdown, spelling, testthat (>= 3.0.0), tidycmprsk (>= 0.2.0), vdiffr (>= 1.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

Language en-US

LazyData true

RoxygenNote 7.2.1

NeedsCompilation no

Author Daniel D. Sjoberg [aut, cre, cph] (<https://orcid.org/0000-0003-0862-2018>), Mark Baillie [aut], Steven Haesendonckx [aut] (<https://orcid.org/0000-0001-8222-3686>), Tim Treis [aut] (<https://orcid.org/0000-0002-9686-4799>)

Maintainer Daniel D. Sjoberg <danield.sjoberg@gmail.com>

Repository CRAN

Date/Publication 2022-10-15 23:12:36 UTC

R topics documented:

`add_censor_mark` *Add Censor Marking*

Description

Add a marking on the figure to represent the time an observations was censored.

Usage

```
add_censor_mark(...)
```

Arguments

`...`	arguments passed to passed to `ggplot2::geom_point(...)` with defaults `shape = 3` and `size = 2`

Value

a ggplot2 figure

Examples

```
survfit2(Surv(time, status) ~ 1, data = df_lung) %>%
  ggsurvfit() +
  add_confidence_interval() +
  add_censor_mark()
```

---

add_confidence_interval

Add Confidence Interval

Description

Add a confidence interval represented by either a ribbon or lines.

Usage

```
add_confidence_interval(type = c("ribbon", "lines"), ...)
```

Arguments

type	string indicating the type of confidence interval to draw. Must be one of `c("ribbon", "lines")`
...	arguments pass to geom. • `type = 'ribbon'`: Defaults are `ggplot2::geom_ribbon(alpha = 0.2, color = NA, ...)` • `type = 'lines'`: Defaults are `ggplot2::geom_step(linetype = "dashed", na.rm = TRUE, ...)`

Value

a ggplot2 figure

Examples

```
survfit2(Surv(time, status) ~ sex, data = df_lung) %>%
  ggsurvfit() +
  add_confidence_interval()

survfit2(Surv(time, status) ~ 1, data = df_lung) %>%
  ggsurvfit() +
  add_confidence_interval(type = "lines")
```

add_legend_title *Add Legend Title*

Description

Add a default or custom title to the figure legend.

Usage

```
add_legend_title(title = NULL)
```

Arguments

title	a string to override the default legend title. Default is NULL

Value

a ggplot2 figure

Examples

```
survfit2(Surv(time, status) ~ surg, data = df_colon) %>%
  ggsurvfit() +
  add_legend_title()
```

add_pvalue *Add p-value*

Description

- add_pvalue("caption"): Add a p-value to the figure via ggplot2::labs(caption=)
- add_pvalue("annotation"): Add a p-value text annotation via ggplot2::annotation("text")

P-values are calculated with survival::survdiff() or tidycmprsk::glance(). Examples of custom placement located in the help file for survfit_p().

When a competing risks figure includes multiple outcomes, only the p-value comparing stratum for the *first* outcome can be placed.

Usage

```
add_pvalue(
  location = c("caption", "annotation"),
  caption = "{p.value}",
  prepend_p = TRUE,
  pvalue_fun = format_p,
  rho = 0,
  ...
)
```

Arguments

location	string indicating where to place p-value. Must be one of c("caption", "annotation")
caption	string to be placed as the caption/annotation. String will be processed with glue::glue(), and the default is "{p.value}"
prepend_p	prepend "p=" to formatted p-value
pvalue_fun	function to round and style p-value with
rho	argument passed to survival::survdiff(rho=)
...	arguments passed to ggplot2::annotate(). Commonly used arguments are x= and y= to place the p-value at the specified coordinates on the plot.

Value

a ggplot2 figure

See Also

survfit_p()

Examples

```
survfit2(Surv(time, status) ~ surg, df_colon) %>%
  ggsurvfit() +
  add_pvalue(caption = "Log-rank {p.value}")

survfit2(Surv(time, status) ~ surg, df_colon) %>%
  ggsurvfit() +
  add_pvalue("annotation", size = 5)
```

---

add_quantile *Add Quantile Annotation*

Description

Add quantile information annotated on to the plot.

Usage

```
add_quantile(y_value = NULL, x_value = NULL, ...)
```

Arguments

y_value, x_value	Numeric value where the line segment will be drawn. Default is y_value=0.5 when both y_value and x_value are unassigned.
...	Named arguments passed to ggplot2::geom_segment() with default linetype = 2

Value

a ggplot2 figure

Examples

```
survfit2(Surv(time, status) ~ sex, data = df_lung) %>%
  ggsurvfit() +
  add_quantile(linetype = 2)

survfit2(Surv(time, status) ~ 1, data = df_lung) %>%
  ggsurvfit() +
  add_quantile(linetype = 2) +
  add_quantile(y_value = 0.9, linetype = 3)

survfit2(Surv(time, status) ~ sex, data = df_lung) %>%
  ggsurvfit() +
  add_quantile(linetype = 2, y_value = NULL, x_value = 10)
```

---

add_risktable *Add risk table*

Description

Add risk tables below the plot showing the number at risk, events observed, and number of censored observations.

Usage

```
add_risktable(
  times = NULL,
  risktable_stats = c("n.risk", "cum.event"),
  risktable_group = c("auto", "strata", "risktable_stats"),
  risktable_height = NULL,
  stats_label = NULL,
  combine_groups = FALSE,
  theme = theme_risktable_default(),
  size = 3.5,
  ...
)
```

Arguments

times	numeric vector of times where risk table values will be placed. Default are the times shown on the x-axis. The times passed here will not modify the tick marks shown on the figure. To modify which tick marks are shown, use `ggplot2::scale_x_continuous(breaks=)`.

`risktable_stats`
: character vector of statistics to show in the risk table. Must be one or more of `c("n.risk", "cum.event", "cum.censor", "n.event", "n.censor")`. Default is `c("n.risk", "cum.event")`.
  - `"n.risk"` Number of patients at risk
  - `"cum.event"` Cumulative number of observed events
  - `"cum.censor"` Cumulative number of censored observations
  - `"n.event"` Number of events in time interval
  - `"n.censor"` Number of censored observations in time interval

`risktable_group`
: String indicating the grouping variable for the risk tables. Default is `"auto"` and will select `"strata"` or `"risktable_stats"` based on context.
  - `"strata"` groups the risk tables per stratum when present.
  - `"risktable_stats"` groups the risk tables per risktable_stats.

`risktable_height`
: A numeric value between 0 and 1 indicates the proportion of the final plot the risk table will occupy.

`stats_label` named vector or list of custom labels. Names are the statistics from `risktable_stats=` and the value is the custom label.

`combine_groups` logical indicating whether to combine the statistics in the risk table across groups. Default is `FALSE`

`theme` A risk table theme. Default is `theme_risktable_default()`

`size, ...` arguments passed to `ggplot2::geom_text(...)`. Pass arguments like, `size = 4` to increase the size of the statistics presented in the table.

Value

a ggplot2 figure

Competing Risks

The `ggcuminc()` can plot multiple competing events. The `"cum.event"` and `"n.event"` statistics are the sum of all events across outcomes *shown on the plot*.

Examples

```
p <-
  survfit2(Surv(time, status) ~ sex, data = df_lung) %>%
  ggsurvfit() +
  add_censor_mark() +
  add_confidence_interval()

p + add_risktable()

p +
  add_risktable(
    risktable_stats = c("n.risk", "cum.event"),
```

```
    stats_label = list(
      cum.event = "Cumulative Observed Events",
      n.risk = "Number at Risk"
    ),
    risktable_group = "strata",
  )

p +
  add_risktable(
    risktable_stats = c("n.risk", "cum.event"),
    combine_groups = TRUE
  )
```

`add_risktable_strata_symbol` *Use Symbol for Strata in Risk Table*

Description

Replace the stratum level names with a color symbol in the risk tables. Use this function when stratum level names are long.

Usage

```
add_risktable_strata_symbol(
  symbol = NULL,
  size = 15,
  face = "bold",
  vjust = 0.3,
  ...
)
```

Arguments

`symbol`	UTF-8 code of shape to replace strata level with. Default is a rectangle (`"\U25AC"`). Other common options are circle (`"\U25CF"`) and diamond (`"\U25C6"`). While a symbol is the most common string to pass here, any string is acceptable.
`size, face, vjust, ...`	arguments passed to a function similar to `ggtext::element_markdown()`

Value

a ggplot2 figure

Examples

```
p <-
  survfit2(Surv(time, status) ~ sex, data = df_lung) %>%
  ggsurvfit(size = 1) +
  add_confidence_interval() +
  add_risktable(risktable_group = "risktable_stats")

 p + add_risktable_strata_symbol()
 p + add_risktable_strata_symbol(symbol = "\U25CF", size = 10)
```

`adtte` *Example phase III clinical trial data set*

Description

Background The example simulated data set is based on large phase III clinical trials in breast cancer such as the ALTTO trial https://ascopubs.org/doi/pdf/10.1200/JCO.2015.62.1797. The example "trial" aims to determine if a combination of two therapies tablemab (T) plus vismab (V) improves outcomes for metastatic human epidermal growth factor 2–positive breast cancer and increases the pathologic complete response in the neoadjuvant setting (i.e. treatment given as a first step to shrink a tumor before the main treatment or surgery).

Usage

```
adtte
```

Format

The data set contains the following variables:

STUDYID The study identifier. A code unique to the clinical trial

SUBJID subject identifier. Numeric ID unique to each patient

USUBJID unique subject identifier. Text ID combining study and patient IDs

AGE age at randomisation (years)

STR01 Hormone receptor status at randomisation

STR01N Hormone receptor positive (Numeric)

STR01L Hormone receptor positive (Long format)

STR02 Prior Radiotherapy at randomisation

STR02N Prior Radiotherapy at randomisation (Numeric)

STR02L Prior Radiotherapy at randomisation (Long format)

TRT01P Planned treatment assigned at randomisation

TRT01PN Planned treatment assigned at randomisation (Numeric)

PARAM Analysis parameter: Progression free survival

PARAMCD Analysis parameter code

AVAL Analysis value (time to event (years)

CNSR Censoring (0 = Event, 1 = Censored)

EVNTDESC Event description

CNSDTDSC Censoring description

DCTREAS Discontinuation from study reason

Details

The trial has four treatment arms, patients with centrally confirmed human epidermal growth factor 2-positive early breast cancer were randomly assigned to 1 year of adjuvant therapy with V, T, their sequence (T to V), or their combination (T+V) for 52 weeks.

The primary end point was progression-free survival (PFS) as defined by Cancer.gov: '"the length of time during and after the treatment of a disease, such as cancer, that a patient lives with the disease but it does not get worse. In a clinical trial, measuring the progression-free survival is one way to see how well a new treatment works"'.

A number of baseline measurements (taken at randomization) are also included such as age, hormone receptor status and prior radiotherapy treatment.

Additional details on reasons for study discontinuation and censoring event description are also included.

The data set adopts an abridged version of the CDISC ADaM ADTTE time to event data model. See here for more info on CDISC ADaM data standards https://www.cdisc.org/standards/foundational/adam and specifically the ADTTE time to event data model here https://www.cdisc.org/standards/foundational/adam/adam-basic-data-structure-bds-time-event-tte-analyses-v1-0.

Source

https://github.com/VIS-SIG/Wonderful-Wednesdays/tree/master/data/2020/2020-04-08

---

df_colon *Formatted Copy of* survival::colon

Description

This is a copy of the colon data set exported by the survival package. This data set, however, has column labels, numeric categorical variables are now factors with assigned levels, and we only include the recurrence outcome.

Usage

```
df_colon
```

Format

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 929 rows and 14 columns.

df_lung *Formatted Copy of* survival::lung

Description

This is a copy of the lung data set exported by the survival package. This data set, however, has column labels and numeric categorical variables are now factors with assigned levels.

Usage

```
df_lung
```

Format

An object of class tbl_df (inherits from tbl, data.frame) with 228 rows and 10 columns.

format_p *Format p-value*

Description

Round and format p-values

Usage

```
format_p(x, digits = 1)
```

Arguments

x	numeric vector of p-values
digits	number of digits large p-values will be rounded to. Default is 2, and must be one of 1, 2, or 3.

Value

a string

Examples

```
p_vec <- c(0.00001, 0.01111, 0.0500000, 0.15, 0.99999)
format_p(p_vec)
format_p(p_vec, 2)
format_p(p_vec, 3)
```

ggcuminc *Plot Cumulative Incidence*

Description

Plot a cumulative incidence object created with `tidycmprsk::cuminc()` or a multi-state object created with `survfit2()`. Read more on multi-state models here.

Usage

```
ggcuminc(
  x,
  outcome = NULL,
  linetype_aes = FALSE,
  theme = theme_ggsurvfit_default(),
  ...
)
```

Arguments

`x`	a 'survfit' object created with `survfit2()`
`outcome`	string indicating which outcome(s) to include in plot. Default is to include the first competing event.
`linetype_aes`	logical indicating whether to add `ggplot2::aes(linetype = strata)` to the `ggplot2::geom_step()` call. When strata are present, the resulting figure will be a mix a various line types for each stratum.
`theme`	a survfit theme. Default is `theme_ggsurvfit_default()`
`...`	arguments passed to `ggplot2::geom_step(...)`, e.g. `size = 2`

Value

a ggplot2 figure

Details

Why do we not use `cmprsk::cuminc()`*?*

The implementation of `cmprsk::cuminc()` does not provide the data required to construct the risk table. Moreover, the `tidycmprsk::cuminc()` has a user-friendly interface making it easy to learn and use.

Examples

```
library(tidycmprsk)

cuminc(Surv(ttdeath, death_cr) ~ trt, trial) %>%
  ggcuminc(outcome = "death from cancer") +
```

```
  add_confidence_interval() +
  add_risktable()

cuminc(Surv(ttdeath, death_cr) ~ trt, trial) %>%
  ggcuminc(outcome = c("death from cancer", "death other causes")) +
  add_risktable()

# using the survival multi-state model
survfit2(Surv(ttdeath, death_cr) ~ trt, trial) %>%
  ggcuminc(outcome = "death from cancer") +
  add_confidence_interval() +
  add_risktable()

survfit2(Surv(ttdeath, death_cr) ~ trt, trial) %>%
  ggcuminc(outcome = c("death from cancer", "death other causes")) +
  add_risktable()
```

`ggsurvfit` *Plot Survival Probability*

Description

Plot survival probabilities (and other transformations) using the results from `survfit2()` or `survival::survfit()`; although, we recommend the former to have the best experience with the **ggsurvfit** package.

Usage

```
ggsurvfit(
  x,
  type = "survival",
  linetype_aes = FALSE,
  theme = theme_ggsurvfit_default(),
  ...
)
```

Arguments

`x`	a 'survfit' object created with `survfit2()`
`type`	type of statistic to report. Available for Kaplan-Meier estimates only. Default is `"survival"`. Must be one of the following:

type	transformation
`"survival"`	`x`
`"risk"`	`1 - x`
`"cumhaz"`	`-log(x)`

`linetype_aes`	logical indicating whether to add `ggplot2::aes(linetype = strata)` to the

	ggplot2::geom_step() call. When strata are present, the resulting figure will be a mix a various line types for each stratum.
theme	a survfit theme. Default is theme_ggsurvfit_default()
...	arguments passed to ggplot2::geom_step(...), e.g. size = 2

Value

a ggplot2 figure

Details

This function creates a ggplot figure from the 'survfit' object. To better understand how to modify the figure, review the simplified code used internally:

```
survfit2(Surv(time, status) ~ sex, data = df_lung) %>%
  tidy_survfit() %>%
  ggplot(aes(x = time, y = estimate, y
             min = conf.low, ymax = conf.low,
             color = strata, fill = strata)) +
  geom_step()
```

Examples

```
# Default publication ready plot
survfit2(Surv(time, status) ~ sex, data = df_lung) %>%
  ggsurvfit()

# Changing statistic type
survfit2(Surv(time, status) ~ sex, data = df_lung) %>%
  ggsurvfit(type = "cumhaz")

# Configuring KM line type to vary by strata
survfit2(Surv(time, status) ~ sex, data = df_lung) %>%
  ggsurvfit(linetype_aes = TRUE)

# Customizing the plot to your needs
survfit2(Surv(time, status) ~ 1, data = df_lung) %>%
  ggsurvfit() +
  add_censor_mark() +
  add_confidence_interval() +
  add_quantile() +
  add_risktable()
```

ggsurvfit_build *Build 'ggsurvfit' Object*

Description

Function takes an object created with `ggsurvfit()` or `ggcuminc()` and prepares the plot for printing. If a plot also has a risk table, this function will build the risk table plots and return them either as list of plots or combined using `patchwork::wrap_plots()`.

This can be particularly useful when you would like to place figures with risk tables side-by-side.

Usage

```
ggsurvfit_build(x, combine_plots = TRUE)
```

Arguments

`x`	an object of class 'ggsurvfit' or 'ggcuminc'
`combine_plots`	logical indicating whether to combine the primary plot and the risk tables. When `TRUE`, plot and risk table(s) are combined with `patchwork::wrap_plots()`. When `FALSE` and the plot has risk tables, they are returned in a list of gtable grobs. Default is `TRUE`.

Value

a list of ggplot2 objects or a single ggplot2 object

Examples

```
# construct plot
p <-
  survfit2(Surv(time, status) ~ surg, df_colon) %>%
  ggsurvfit() +
  add_risktable() +
  scale_y_continuous(limits = c(0, 1))

# build plots
built_p <- ggsurvfit_build(p, combine_plots = FALSE)

# reconstruct original figure print with risktables
patchwork::wrap_plots(
  built_p[[1]],
  built_p[[2]],
  built_p[[3]],
  ncol = 1,
  heights = c(0.70, 0.15, 0.15)
)

# place plots side-by-side (plots must be built before placement with patchwork)
```

```
patchwork::wrap_plots(
  ggsurvfit_build(p),
  ggsurvfit_build(p),
  ncol = 2
)
```

`scale_ggsurvfit` *Apply Scales*

Description

The most common figure created with this package is a survival curve. This scale applies modifications often seen in these figures.

- `scale_y_continuous(expand = c(0.025, 0), limits = c(0, 1), label = scales::label_percent())`.
- `scale_x_continuous(expand = c(0.015, 0), n.breaks = 8)`

If you use this function, you **must** include **all** scale specifications that would appear in `scale_x_continuous()` or `scale_y_continuous()`. For example, it's common you'll need to specify the x-axis break points. `scale_ggsurvfit(x_scales=list(breaks=0:9))`.

To reset any of the above settings to their ggplot2 default, set the value to `NULL`, e.g. `y_scales = list(limits = NULL)`.

Usage

```
scale_ggsurvfit(x_scales = list(), y_scales = list())
```

Arguments

`x_scales`	a named list of arguments that will be passed to `ggplot2::scale_x_continuous()`.
`y_scales`	a named list of arguments that will be passed to `ggplot2::scale_y_continuous()`.

Value

a ggplot2 figure

Examples

```
ggsurvfit <-
  survfit2(Surv(time, status) ~ surg, data = df_colon) %>%
  ggsurvfit(size = 1) +
  add_confidence_interval()

# use the function defaults
ggsurvfit + scale_ggsurvfit()

# specify additional scales
ggsurvfit +
  scale_ggsurvfit(x_scales = list(breaks = seq(0, 8, by = 2)))
```

stepribbon *Step ribbon statistic*

Description

Provides stairstep values for ribbon plots

Usage

```
stat_stepribbon(
  mapping = NULL,
  data = NULL,
  geom = "ribbon",
  position = "identity",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE,
  direction = "hv",
  ...
)

StatStepribbon
```

Arguments

mapping	Set of aesthetic mappings created by aes() or aes_(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
geom	which geom to use; defaults to "ribbon"
position	Position adjustment, either as a string, or the result of a call to a position adjustment function.
na.rm	If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.
show.legend	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.

inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. borders().
direction	hv for horizontal-vertical steps, vh for vertical-horizontal steps
...	Other arguments passed on to layer(). These are often aesthetics, used to set an aesthetic to a fixed value, like colour = "red" or size = 3. They may also be parameters to the paired geom/stat.

Format

An object of class StatStepRibbon (inherits from Stat, ggproto, gg) of length 3.

Value

a ggplot2 figure

References

https://groups.google.com/forum/?fromgroups=#!topic/ggplot2/9cFWHaH1CPs

Examples

```
survfit(Surv(time, status) ~ 1, data = df_lung) %>%
  survival::survfit0() %>%
  broom::tidy() %>%
  ggplot(aes(x = time, y = estimate, ymin = conf.low, ymax = conf.high)) +
  geom_step() +
  geom_ribbon(stat = "stepribbon", alpha = 0.2)
```

---

survfit2 *Create survival curves*

Description

Simple wrapper for survival::survfit.formula() except the environment is also included in the returned object.

Use this function with all other functions in this package to ensure all elements are calculable.

Usage

```
survfit2(formula, ...)
```

Arguments

`formula` a formula object, which must have a `Surv` object as the response on the left of the ~ operator and, if desired, terms separated by + operators on the right. One of the terms may be a `strata` object. For a single survival curve the right hand side should be ~ `1`.

`...` Arguments passed on to `survival::survfit.formula`

`data` a data frame in which to interpret the variables named in the formula, `subset` and `weights` arguments.

`weights` The weights must be nonnegative and it is strongly recommended that they be strictly positive, since zero weights are ambiguous, compared to use of the `subset` argument.

`subset` expression saying that only a subset of the rows of the data should be used in the fit.

`na.action` a missing-data filter function, applied to the model frame, after any `subset` argument has been used. Default is `options()$na.action`.

`stype` the method to be used estimation of the survival curve: 1 = direct, 2 = exp(cumulative hazard).

`ctype` the method to be used for estimation of the cumulative hazard: 1 = Nelson-Aalen formula, 2 = Fleming-Harrington correction for tied events.

`id` identifies individual subjects, when a given person can have multiple lines of data.

`cluster` used to group observations for the infinitesimal jackknife variance estimate, defaults to the value of id.

`robust` logical, should the function compute a robust variance. For multi-state survival curves this is true by default. For single state data see details, below.

`istate` for multi-state models, identifies the initial state of each subject or observation

`timefix` process times through the `aeqSurv` function to eliminate potential roundoff issues.

`etype` a variable giving the type of event. This has been superseded by multi-state Surv objects and is deprecated; see example below.

`error` this argument is no longer used

Value

survfit2 object

`survfit2()` vs `survfit()`

Both functions have identical inputs, so why do we need `survfit2()`?

The *only* difference between `survfit2()` and `survival::survfit()` is that the former tracks the environment from which the call to the function was made.

The definition of `survfit2()` is unremarkably simple:

```
survfit2 <- function(formula, ...) {
  # construct survfit object
  survfit <- survival::survfit(formula, ...)

  # add the environment
  survfit$.Environment = <calling environment>

  # add class and return
  class(survfit) <- c("survfit2", "survfit")
  survfit
}
```

The environment is needed to ensure the survfit call can be accurately reconstructed or parsed at any point post estimation. The call is parsed when p-values are reported and when labels are created. For example, the raw variable names appear in the output of a stratified `survfit()` result, e.g. `"sex=Female"`. When using `survfit2()`, the originating data frame and formula may be parsed and the raw variable names removed.

Most functions in the package work with both `survfit2()` and `survfit()`; however, the output will be styled in a preferable format with `survfit2()`.

See Also

`survival::survfit.formula()`

Examples

```
# With `survfit()`
fit <- survfit(Surv(time, status) ~ sex, data = df_lung)
fit

# With `survfit2()`
fit2 <- survfit2(Surv(time, status) ~ sex, data = df_lung)
fit2

# Consistent behavior with other functions
summary(fit, times = c(10, 20))

summary(fit2, times = c(10, 20))
```

---

survfit2_p *Calculate p-value*

Description

The function `survfit2_p()` wraps `survival::survdiff()` and returns a formatted p-value.

Usage

```
survfit2_p(x, pvalue_fun = format_p, prepend_p = TRUE, rho = 0)
```

Arguments

x	a 'survfit2' object
pvalue_fun	function to round and style p-value with
prepend_p	prepend "p=" to formatted p-value
rho	argument passed to `survival::survdiff(rho=)`

Value

a string

Examples

```
sf <- survfit2(Surv(time, status) ~ sex, data = df_lung)

sf %>%
  ggsurvfit() +
  add_confidence_interval() +
  add_risktable() +
  labs(caption = glue::glue("Log-rank {survfit2_p(sf)}"))

sf %>%
  ggsurvfit() +
  add_confidence_interval() +
  add_risktable() +
  annotate("text", x = 2, y = 0.05, label = glue::glue("{survfit2_p(sf)}"))
```

Surv_CNSR *Create a Survival Outcome from CDISC Data*

Description

The aim of `Surv_CNSR()` is to map the inconsistency in data convention between the survival package and CDISC ADaM ADTTE data model.

The function creates a survival object (e.g. `survival::Surv()`) that uses CDISC ADaM ADTTE coding conventions and converts the arguments to the status/event variable convention used in the survival package.

The `AVAL` and `CNSR` arguments are passed to `survival::Surv(time = AVAL, event = 1 - CNSR, type = "right", origin = 0)`.

Usage

```
Surv_CNSR(AVAL, CNSR)
```

Arguments

AVAL	The follow-up time. The follow-up time is assumed to originate from zero. When no argument is passed, the default value is a column/vector named AVAL.
CNSR	The censoring indicator where 1=censored and 0=death/event. When no argument is passed, the default value is a column/vector named CNSR.

Value

Object of class 'Surv'

Details

The Surv_CNSR() function creates a survival object utilizing the expected data structure in the CDISC ADaM ADTTE data model, mapping the CDISC ADaM ADTTE coding conventions with the expected status/event variable convention used in the survival package—specifically, the coding convention used for the status/event indicator. The survival package expects the status/event indicator in the following format: 0=alive, 1=dead. Other accepted choices are TRUE/FALSE (TRUE = death) or 1/2 (2=death). A final but risky option is to omit the indicator variable, in which case all subjects are assumed to have an event.

The CDISC ADaM ADTTE data model adopts a different coding convention for the event/status indicator. Using this convention, the event/status variable is named 'CNSR' and uses the following coding: censor = 1, status/event = 0.

See Also

survival::Surv()

Examples

```
# Use the `Surv_CNSR()` function with ggsurvfit functions
survfit2(formula = Surv_CNSR() ~ STR01, data = adtte) %>%
  ggsurvfit() +
  add_confidence_interval()

# Use the `Surv_CNSR()` function with functions from other packages as well
survival::survfit(Surv_CNSR() ~ STR01, data = adtte)
survival::survreg(Surv_CNSR() ~ STR01 + AGE, data = adtte) %>%
  broom::tidy()
```

---

theme_ggsurvfit *Survfit Plot Themes*

Description

Returns ggplot list of calls defining a theme.

- theme_ggsurvfit_default(): Builds on theme_bw() with increased text sizes.
- theme_ggsurvfit_KMunicate(): Theme to create KMunicate-styled figures. doi:10.1136/bmjopen2019030215

Usage

```
theme_ggsurvfit_default()

theme_ggsurvfit_KMunicate()
```

Value

a ggplot2 theme

Examples

```
survfit2(Surv(time, status) ~ sex, data = df_lung) %>%
  ggsurvfit(theme = theme_ggsurvfit_default())
```

---

theme_risktable *Risk Table Themes*

Description

Returns ggplot list of calls defining a theme meant to be applied to a risk table.

Usage

```
theme_risktable_default(axis.text.y.size = 10, plot.title.size = 10.75)

theme_risktable_boxed(axis.text.y.size = 10, plot.title.size = 10.75)
```

Arguments

`axis.text.y.size`
: text size of the labels on the left of the risk table

`plot.title.size`
: text size of the risk table title

Value

a ggplot2 figure

Examples

```
p <- survfit2(Surv(time, status) ~ 1, data = df_lung) %>% ggsurvfit()

# default -------------------------------------
p + add_risktable(theme = theme_risktable_default())

# larger text ---------------------------------
p +
  add_risktable(
    size = 4,
```

```
    theme = theme_risktable_default(axis.text.y.size = 12,
                                    plot.title.size = 14)
  )

# boxed ----------------------------------------
p + add_risktable(theme = theme_risktable_boxed())

# none -----------------------------------------
p + add_risktable(theme = NULL, risktable_height = 0.20)
```

`tidy_cuminc` *Tidy a cuminc object*

Description

The tidycmprsk package exports a tidier for `"cuminc"` objects. This function adds on top of that and returns more information.

Usage

```
tidy_cuminc(x, times = NULL)
```

Arguments

`x`	a 'cuminc' object created with `tidycmprsk::cuminc()`
`times`	numeric vector of times. Default is `NULL`, which returns all observed times.

Value

a tibble

Examples

```
library(tidycmprsk)

cuminc(Surv(ttdeath, death_cr) ~ trt, trial) %>%
  tidy_cuminc()
```

`tidy_survfit` *Tidy a survfit object*

Description

The broom package exports a tidier for `"survfit"` objects. This function adds on top of that and returns more information. The function also utilizes additional information stored when the survfit object is created with `survfit2()`. It's recommended to always use this function with `survfit2()`.

Usage

```
tidy_survfit(x, times = NULL, type = c("survival", "risk", "cumhaz"))
```

Arguments

`x`	a 'survfit' object created with `survfit2()`
`times`	numeric vector of times. Default is `NULL`, which returns all observed times.
`type`	type of statistic to report. Available for Kaplan-Meier estimates only. Default is `"survival"`. Must be one of the following:

type	transformation
`"survival"`	`x`
`"risk"`	`1 - x`
`"cumhaz"`	`-log(x)`

Value

a tibble

Examples

```
survfit2(Surv(time, status) ~ factor(ph.ecog), data = df_lung) %>%
  tidy_survfit()
```

Index