Package 'survminer'

March 9, 2021

Type Package

Title Drawing Survival Curves using 'ggplot2'

Version 0.4.9

Date 2021-03-09

Description Contains the function 'ggsurvplot()' for drawing easily beautiful and 'ready-to-publish' survival curves with the 'number at risk' table and 'censoring count plot'. Other functions are also available to plot adjusted curves for `Cox` model and to visually examine 'Cox' model assumptions.

License GPL-2

LazyData TRUE

Encoding UTF-8

Depends ggplot2, ggpubr(>= 0.1.6)

Imports grid, gridExtra (>= 2.0), magrittr, maxstat, methods, scales, survival, stats, broom, dplyr, tidyr, survMisc, purrr, tibble, rlang, ggtext (>= 0.1.0)

Suggests knitr, flexsurv, cmprsk, markdown, testthat, rmarkdown

VignetteBuilder knitr

URL https://rpkgs.datanovia.com/survminer/index.html

BugReports https://github.com/kassambara/survminer/issues

RoxygenNote 7.1.1

Collate 'BMT.R' 'BRCAOV.survInfo.R' 'add_ggsurvplot.R' 'utilities.R' 'surv_summary.R' 'ggsurvtable.R' 'surv_pvalue.R' 'ggsurvplot_df.R' 'ggsurvplot_core.R' 'ggsurvplot_add_all.R' 'ggsurvplot_list.R' 'ggsurvplot_group_by.R' 'ggsurvplot.R' 'arrange_ggsurvplots.R' 'ggadjustedcurves.R' 'ggcompetingrisks.R' 'ggcoxdiagnostics.R' 'ggcoxfunctional.R' 'ggcoxzph.R' 'ggflexsurvplot.R' 'ggforest.R' 'ggsurvevents.R' 'ggsurvplot_combine.R' 'ggsurvplot_facet.R' 'ggsurvtheme.R' 'ggurvplot_arguments.R' 'myeloma.R' 'pairwise_survdiff.R' 'surv_cutpoint.R' 'surv_group_by.R' 'surv_fit.R' 'surv_median.R'

NeedsCompilation no

Author Alboukadel Kassambara [aut, cre],
Marcin Kosinski [aut],
Przemyslaw Biecek [aut],
Scheipl Fabian [ctb]

Maintainer Alboukadel Kassambara <alboukadel.kassambara@gmail.com>

Repository CRAN

Date/Publication 2021-03-09 09:50:03 UTC

R topics documented:

add_ggsurvplot *Add Components to a ggsurvplot*

Description

Allows to add ggplot components - theme(), labs(), ... - to an object of class ggsurv, which is a list of ggplots.

Usage

```
## S3 method for class 'ggsurv'
e1 + e2

e1 %++% e2
```

Arguments

e1	an object of class ggsurv.
e2	a plot component such as theme and labs.

See Also

theme_survminer and ggsurvplot

Examples

```
# Fit survival curves
require("survival")
fit<- survfit(Surv(time, status) ~ sex, data = lung)

# Basic survival curves
p <- ggsurvplot(fit, data = lung, risk.table = TRUE,
   main = "Survival curve",
   submain = "Based on Kaplan-Meier estimates",
   caption = "created with survminer"
   )
p

# Customizing the plots
p + theme_survminer(
     font.main = c(16, "bold", "darkblue"),
     font.submain = c(15, "bold.italic", "purple"),
     font.caption = c(14, "plain", "orange"),
     font.x = c(14, "bold.italic", "red"),
     font.y = c(14, "bold.italic", "darkred"),
     font.tickslab = c(12, "plain", "darkgreen")
)
```

`arrange_ggsurvplots` *Arranging Multiple ggsurvplots*

Description

Arranging multiple ggsurvplots on the same page.

Usage

```
arrange_ggsurvplots(
  x,
  print = TRUE,
  title = NA,
  ncol = 2,
  nrow = 1,
  surv.plot.height = NULL,
  risk.table.height = NULL,
  ncensor.plot.height = NULL,
  ...
)
```

Arguments

`x`	a list of ggsurvplots.
`print`	logical value. If TRUE, the arranged plots are displayed.
`title`	character vector specifying page title. Default is NA.
`ncol, nrow`	the number of columns and rows, respectively.
`surv.plot.height`	the height of the survival plot on the grid. Default is 0.75. Ignored when risk.table = FALSE. `1-risk.table.height -ncensor.plot.height` when `risk.table = TRUE` and `ncensor.plot = TRUE`
`risk.table.height`	the height of the risk table on the grid. Increase the value when you have many strata. Default is 0.25. Ignored when risk.table = FALSE.
`ncensor.plot.height`	The height of the censor plot. Used when `ncensor.plot = TRUE`.
`...`	not used

Value

returns an invisible object of class arrangelist (see marrangeGrob), which can be saved into a pdf file using the function ggsave.

Author(s)

Alboukadel Kassambara, <alboukadel.kassambara@gmail.com>

Examples

```
# Fit survival curves
require("survival")
fit<- survfit(Surv(time, status) ~ sex, data = lung)

# List of ggsurvplots
require("survminer")
splots <- list()
splots[[1]] <- ggsurvplot(fit, data = lung, risk.table = TRUE, ggtheme = theme_minimal())
splots[[2]] <- ggsurvplot(fit, data = lung, risk.table = TRUE, ggtheme = theme_grey())

# Arrange multiple ggsurvplots and print the output
arrange_ggsurvplots(splots, print = TRUE,
  ncol = 2, nrow = 1, risk.table.height = 0.4)

## Not run:
# Arrange and save into pdf file
res <- arrange_ggsurvplots(splots, print = FALSE)
ggsave("myfile.pdf", res)

## End(Not run)
```

BMT *Bone Marrow Transplant*

Description

Bone marrow transplant data from L Scrucca et aL., Bone Marrow Transplantation (2007). Data from 35 patients with acute leukaemia who underwent HSCT. Used for competing risk analysis.

Usage

```
data("BMT")
```

Format

A data frame with 35 rows and 3 columns.

- dis: disease; 0 = ALL; 1 = AML
- ftime: follow-up time
- status: 0 = censored (survival); 1 = Transplant-related mortality; 2 = relapse

References

Scrucca L, Santucci A, Aversa F. Competing risk analysis using R: an easy guide for clinicians. Bone Marrow Transplant. 2007 Aug;40(4):381-7.

Examples

```
data(BMT)
if(require("cmprsk")){

# Data preparaion
#++++++++++++++++++++
# Label diseases
BMT$dis <- factor(BMT$dis, levels = c(0,1),
   labels = c("ALL", "AML"))
# Label status
BMT$status <- factor(BMT$status, levels = c(0,1,2),
  labels = c("Censored","Mortality","Relapse"))

# Cumulative Incidence Function
# ++++++++++++++++++++++++++
fit <- cmprsk::cuminc(
  ftime = BMT$ftime,      # Failure time variable
  fstatus = BMT$status,   # Codes for different causes of failure
  group = BMT$dis         # Estimates will calculated within groups
 )

# Visualize
# ++++++++++++++++++++++
ggcompetingrisks(fit)
ggcompetingrisks(fit, multiple_panels = FALSE,
  legend = "right")

}
```

`BRCAOV.survInfo` *Breast and Ovarian Cancers Survival Information*

Description

Breat and Ovarian cancers survival information from the RTCGA.clinical R/Bioconductor package.http://rtcga.github.io/RTCGA/.

Usage

```
data("BRCAOV.survInfo")
```

Format

A data frame with 1674 rows and 4 columns.

- times: follow-up time;
- bcr_patient_barcode: Patient bar code;
- patient.vital_status = survival status. 0 = alive, 1 = dead;

- admin.disease_code: disease code. brca = breast cancer, ov = ovarian cancer.

Source

From the RTCGA.clinical R/Bioconductor package. The data is generated as follow:

```
# Installing RTCGA.clinical
source("https://bioconductor.org/biocLite.R")
biocLite("RTCGA.clinical")

# Generating the BRCAOV survival information
library(RTCGA.clinical)
survivalTCGA(BRCA.clinical, OV.clinical,
extract.cols = "admin.disease_code") -> BRCAOV.survInfo
```

Examples

```
data(BRCAOV.survInfo)
library(survival)
fit <- survfit(Surv(times, patient.vital_status) ~ admin.disease_code,
           data = BRCAOV.survInfo)
ggsurvplot(fit, data = BRCAOV.survInfo, risk.table = TRUE)
```

---

ggadjustedcurves *Adjusted Survival Curves for Cox Proportional Hazards Model*

Description

The function `surv_adjustedcurves()` calculates while the function `ggadjustedcurves()` plots adjusted survival curves for the `coxph` model. The main idea behind this function is to present expected survival curves calculated based on Cox model separately for subpopulations. The very detailed description and interesting discussion of adjusted curves is presented in 'Adjusted Survival Curves' by Terry Therneau, Cynthia Crowson, Elizabeth Atkinson (2015) `https://cran.r-project.org/web/packages/s` Many approaches are discussed in this article. Currently four approaches (two unbalanced, one conditional and one marginal) are implemented in the `ggadjustedcurves()` function. See the section Details.

Usage

```
ggadjustedcurves(
  fit,
  variable = NULL,
  data = NULL,
  reference = NULL,
  method = "conditional",
  fun = NULL,
```

```
  palette = "hue",
  ylab = "Survival rate",
  size = 1,
  ggtheme = theme_survminer(),
  ...
)

surv_adjustedcurves(
  fit,
  variable = NULL,
  data = NULL,
  reference = NULL,
  method = "conditional",
  size = 1,
  ...
)
```

Arguments

`fit`	an object of class coxph.object - created with coxph function.
`variable`	a character, name of the grouping variable to be plotted. If not supplied then it will be extracted from the model formula from the `strata()` component. If there is no `strata()` component then only a single curve will be plotted - average for the thole population.
`data`	a dataset for predictions. If not supplied then data will be extracted from the `fit` object.
`reference`	a dataset for reference population, to which dependent variables should be balanced. If not specified, then the `data` will be used instead. Note that the `reference` dataset should contain all variables used in `fit` object.
`method`	a character, describes how the expected survival curves shall be calculated. Possible options: 'single' (average for population), 'average' (averages for subpopulations), 'marginal', 'conditional' (averages for subpopulations after rebalancing). See the Details section for further information.
`fun`	an arbitrary function defining a transformation of the survival curve. Often used transformations can be specified with a character argument: "event" plots cumulative events (f(y) = 1-y), "cumhaz" plots the cumulative hazard function (f(y) = -log(y)), and "pct" for survival probability in percentage.
`palette`	the color palette to be used. Allowed values include "hue" for the default hue color scale; "grey" for grey color palettes; brewer palettes e.g. "RdBu", "Blues", ...; or custom color palette e.g. c("blue", "red"); and scientific journal palettes from ggsci R package, e.g.: "npg", "aaas", "lancet", "jco", "ucscgb", "uchicago", "simpsons" and "rickandmorty". See details section for more information. Can be also a numeric vector of length(groups); in this case a basic color palette is created using the function palette.
`ylab`	a label for oy axis.
`size`	the curve size.

ggtheme	function, ggplot2 theme name. Allowed values include ggplot2 official themes: see theme.
...	further arguments passed to the function ggpar for customizing the plot.

Details

Currently four approaches are implemented in the ggadjustedcurves() function.

For method = "single" a single survival curve is calculated and plotted. The curve presents an expected survival calculated for population data calculated based on the Cox model fit.

For method = "average" a separate survival curve is plotted for each level of a variable listed as variable. If this argument is not specified, then it will be extracted from the strata component of fit argument. Each curve presents an expected survival calculated for subpopulation from data based on a Cox model fit. Note that in this method subpopulations are NOT balanced.

For method = "marginal" a survival curve is plotted for each level of a grouping variable selected by variable argument. If this argument is not specified, then it will be extracted from the strata component of fit object. Subpopulations are balanced with respect to variables in the fit formula to keep distributions similar to these in the reference population. If no reference population is specified, then the whole data is used as a reference population instead. The balancing is performed in a following way: (1) for each subpopulation a logistic regression model is created to model the odds of being in the subpopulation against the reference population given the other variables listed in a fit object, (2) reverse probabilities of belonging to a specified subpopulation are used as weights in the Cox model, (3) the Cox model is refitted with weights taken into account, (4) expected survival curves are calculated for each subpopulation based on a refitted weighted model.

For method = "conditional" a separate survival curve is plotted for each level of a grouping variable selected by variable argument. If this argument is not specified, then it will be extracted from the strata component of fit object. Subpopulations are balanced in a following way: (1) the data is replicated as many times as many subpopulations are considered (say k), (2) for each row in original data a set of k copies are created and for every copy a different value of a grouping variable is assigned, this will create a new dataset balanced in terms of grouping variables, (3) expected survival is calculated for each subpopulation based on the new artificial dataset. Here the model fit is not refitted.

Note that surv_adjustedcurves function calculates survival curves and based on this function one can calculate median survival.

Value

Returns an object of class gg.

Author(s)

Przemyslaw Biecek, <przemyslaw.biecek@gmail.com>

Examples

```
library(survival)
fit2 <- coxph( Surv(stop, event) ~ size, data = bladder )
# single curve
```

```
ggadjustedcurves(fit2, data = bladder)
curve <- surv_adjustedcurves(fit2, data = bladder)

fit2 <- coxph( Surv(stop, event) ~ size + strata(rx), data = bladder )
# average in groups
ggadjustedcurves(fit2, data = bladder, method = "average", variable = "rx")
curve <- surv_adjustedcurves(fit2, data = bladder, method = "average", variable = "rx")

# conditional balancing in groups
ggadjustedcurves(fit2, data = bladder, method = "marginal", variable = "rx")
curve <- surv_adjustedcurves(fit2, data = bladder, method = "marginal", variable = "rx")

# selected reference population
ggadjustedcurves(fit2, data = bladder, method = "marginal", variable = "rx",
    reference = bladder[bladder$rx == "1",])

# conditional balancing in groups
ggadjustedcurves(fit2, data = bladder, method = "conditional", variable = "rx")
curve <- surv_adjustedcurves(fit2, data = bladder, method = "conditional", variable = "rx")

## Not run:
# this will take some time
fdata <- flchain[flchain$futime >=7,]
fdata$age2 <- cut(fdata$age, c(0,54, 59,64, 69,74,79, 89, 110),
                  labels = c(paste(c(50,55,60,65,70,75,80),
                                   c(54,59,64,69,74,79,89), sep='-'), "90+"))
fdata$group <- factor(1+ 1*(fdata$flc.grp >7) + 1*(fdata$flc.grp >9),
                      levels=1:3,
                      labels=c("FLC < 3.38", "3.38 - 4.71", "FLC > 4.71"))
# single curve
fit <- coxph( Surv(futime, death) ~ age*sex, data = fdata)
ggadjustedcurves(fit, data = fdata, method = "single")

# average in groups
fit <- coxph( Surv(futime, death) ~ age*sex + strata(group), data = fdata)
ggadjustedcurves(fit, data = fdata, method = "average")

# conditional balancing in groups
ggadjustedcurves(fit, data = fdata, method = "conditional")

# marginal balancing in groups
ggadjustedcurves(fit, data = fdata, method = "marginal", reference = fdata)

## End(Not run)
```

---

ggcompetingrisks *Cumulative Incidence Curves for Competing Risks*

Description

This function plots Cumulative Incidence Curves. For cuminc objects it's a ggplot2 version of plot.cuminc. For survfitms objects a different geometry is used, as suggested by @teigentler.

Usage

```
ggcompetingrisks(
  fit,
  gnames = NULL,
  gsep = " ",
  multiple_panels = TRUE,
  ggtheme = theme_survminer(),
  coef = 1.96,
  conf.int = FALSE,
  ...
)
```

Arguments

fit	an object of a class cmprsk::cuminc - created with cmprsk::cuminc function or survfitms created with survfit function.
gnames	a vector with group names. If not supplied then will be extracted from fit object (cuminc only).
gsep	a separator that extracts group names and event names from gnames object (cuminc only).
multiple_panels	if TRUE then groups will be plotted in different panels (cuminc only).
ggtheme	function, ggplot2 theme name. Default value is theme_survminer. Allowed values include ggplot2 official themes: see theme.
coef	see conf.int, scaling actor for the ribbon. The default value is 1.96.
conf.int	if TRUE then additional layer (geom_ribbon) is added around the point estimate. The ribon is plotted with boundries +- coef*standard deviation.
...	further arguments passed to the function ggpar for customizing the plot.

Value

Returns an object of class gg.

Author(s)

Przemyslaw Biecek, <przemyslaw.biecek@gmail.com>

Examples

```
## Not run:
if(require("cmprsk")){
```

```
set.seed(2)
ss <- rexp(100)
gg <- factor(sample(1:3,100,replace=TRUE),1:3,c('BRCA','LUNG','OV'))
cc <- factor(sample(0:2,100,replace=TRUE),0:2,c('no event', 'death', 'progression'))
strt <- sample(1:2,100,replace=TRUE)

# handles cuminc objects
print(fit <- cmprsk::cuminc(ss,cc,gg,strt))
ggcompetingrisks(fit)
ggcompetingrisks(fit, multiple_panels = FALSE)
ggcompetingrisks(fit, conf.int = TRUE)
ggcompetingrisks(fit, multiple_panels = FALSE, conf.int = TRUE)

# handles survfitms objects
library(survival)
df <- data.frame(time = ss, group = gg, status = cc, strt)
fit2 <- survfit(Surv(time, status, type="mstate") ~ 1, data=df)
ggcompetingrisks(fit2)
fit3 <- survfit(Surv(time, status, type="mstate") ~ group, data=df)
ggcompetingrisks(fit3)
}

  library(ggsci)
  library(cowplot)
  ggcompetingrisks(fit3) + theme_cowplot() + scale_fill_jco()

## End(Not run)
```

---

ggcoxdiagnostics *Diagnostic Plots for Cox Proportional Hazards Model with ggplot2*

Description

Displays diagnostics graphs presenting goodness of Cox Proportional Hazards Model fit, that can be calculated with coxph function.

Usage

```
ggcoxdiagnostics(
  fit,
  type = c("martingale", "deviance", "score", "schoenfeld", "dfbeta", "dfbetas",
    "scaledsch", "partial"),
  ...,
  linear.predictions = type %in% c("martingale", "deviance"),
  ox.scale = ifelse(linear.predictions, "linear.predictions", "observation.id"),
  hline = TRUE,
  sline = TRUE,
  sline.se = TRUE,
  hline.col = "red",
```

```
  hline.size = 1,
  hline.alpha = 1,
  hline.yintercept = 0,
  hline.lty = "dashed",
  sline.col = "blue",
  sline.size = 1,
  sline.alpha = 0.3,
  sline.lty = "dashed",
  point.col = "black",
  point.size = 1,
  point.shape = 19,
  point.alpha = 1,
  title = NULL,
  subtitle = NULL,
  caption = NULL,
  ggtheme = ggplot2::theme_bw()
)
```

Arguments

`fit`	an object of class coxph.object - created with coxph function.
`type`	the type of residuals to present on Y axis of a diagnostic plot. The same as in residuals.coxph: character string indicating the type of residual desired. Possible values are `"martingale"`,`"deviance"`,`"score"`,`"schoenfeld"`,`"dfbeta"`,`"dfbetas"` and `"scaledsch"`. Only enough of the string to determine a unique match is required.
`...`	further arguments passed to `residuals.coxph` or to the function ggpar for customizing the plot.
`linear.predictions`	(deprecated, see `ox.scale`) a logical value indicating whether to show linear predictions for observations (`TRUE`) or just indexed of observations (`FALSE`) on X axis.
`ox.scale`	one value from `c("linear.predictions","observation.id","time")`. It defines what will be presented on OX scale. Possible values: y hat for `"linear.predictions"`, Id of an observation for `"observation.id"` or Time for `"time"`.
`hline`	a logical - should the horizontal line be added to highlight the `Y=0` level.
`sline, sline.se`	a logical - should the smooth line be added to highlight the local average for residuals.
`hline.col, hline.size, hline.lty, hline.alpha, hline.yintercept`	color, size, linetype, visibility and Y-axis coordinate to be used for geom_hline. Used only when `hline = TRUE`.
`sline.col, sline.size, sline.lty, sline.alpha`	color, size, linetype and visibility to be used for geom_smooth. Used only when `sline = TRUE`.
`point.col, point.size, point.shape, point.alpha`	color, size, shape and visibility to be used for points.

`title, subtitle, caption`
: main title, subtitle and caption.

`ggtheme`
: function, ggplot2 theme name. Default value is ggplot2::theme_bw(). Allowed values include ggplot2 official themes: see `theme`.

Value

Returns an object of class `ggplot`.

Functions

- `ggcoxdiagnostics`: Diagnostic Plots for Cox Proportional Hazards Model with **ggplot2**

Author(s)

Marcin Kosinski , <m.p.kosinski@gmail.com>

Examples

```
library(survival)
coxph.fit2 <- coxph(Surv(futime, fustat) ~ age + ecog.ps, data=ovarian)
ggcoxdiagnostics(coxph.fit2, type = "deviance")

ggcoxdiagnostics(coxph.fit2, type = "schoenfeld", title = "Diagnostic plot")
ggcoxdiagnostics(coxph.fit2, type = "deviance", ox.scale = "time")
ggcoxdiagnostics(coxph.fit2, type = "schoenfeld", ox.scale = "time",
                 title = "Diagnostic plot", subtitle = "Data comes from survey XYZ",
                 font.subtitle = 9)
ggcoxdiagnostics(coxph.fit2, type = "deviance", ox.scale = "linear.predictions",
                 caption = "Code is available here - link", font.caption = 10)
ggcoxdiagnostics(coxph.fit2, type = "schoenfeld", ox.scale = "observation.id")
ggcoxdiagnostics(coxph.fit2, type = "scaledsch", ox.scale = "time")
```

---

ggcoxfunctional *Functional Form of Continuous Variable in Cox Proportional Hazards Model*

Description

Displays graphs of continuous explanatory variable against martingale residuals of null cox proportional hazards model, for each term in of the right side of `formula`. This might help to properly choose the functional form of continuous variable in cox model (coxph). Fitted lines with lowess function should be linear to satisfy cox proportional hazards model assumptions.

Usage

```
ggcoxfunctional(
  formula,
  data = NULL,
  fit,
  iter = 0,
  f = 0.6,
  point.col = "red",
  point.size = 1,
  point.shape = 19,
  point.alpha = 1,
  xlim = NULL,
  ylim = NULL,
  ylab = "Martingale Residuals \nof Null Cox Model",
  title = NULL,
  caption = NULL,
  ggtheme = theme_survminer(),
  ...
)

## S3 method for class 'ggcoxfunctional'
print(x, ..., newpage = TRUE)
```

Arguments

formula	a formula object, with the response on the left of a ~ operator, and the terms on the right. The response must be a survival object as returned by the Surv function.
data	a data.frame in which to interpret the variables named in the formula,
fit	an object of class coxph.object - created with coxph function.
iter	parameter of lowess.
f	parameter of lowess.
point.col, point.size, point.shape, point.alpha	color, size, shape and visibility to be used for points.
xlim, ylim	x and y axis limits e.g. xlim = c(0, 1000), ylim = c(0, 1).
ylab	y axis label.
title	the title of the final grob (top in arrangeGrob)
caption	the caption of the final grob (bottom in arrangeGrob)
ggtheme	function, ggplot2 theme name. Allowed values include ggplot2 official themes: see theme.
...	further arguments passed to the function ggpar for customizing the plot.
x	an object of class ggcoxfunctional
newpage	open a new page. See grid.arrange.

Value

Returns an object of class ggcoxfunctional which is a list of ggplots.

Functions

- ggcoxfunctional: Functional Form of Continuous Variable in Cox Proportional Hazards Model.

Author(s)

Marcin Kosinski , <m.p.kosinski@gmail.com>

Examples

```
library(survival)
data(mgus)
res.cox <- coxph(Surv(futime, death) ~ mspike + log(mspike) + I(mspike^2) +
    age + I(log(age)^2) + I(sqrt(age)), data = mgus)
ggcoxfunctional(res.cox,  data = mgus, point.col = "blue", point.alpha = 0.5)
ggcoxfunctional(res.cox, data = mgus, point.col = "blue", point.alpha = 0.5,
                title = "Pass the title", caption = "Pass the caption")
```

---

ggcoxzph *Graphical Test of Proportional Hazards with ggplot2*

Description

Displays a graph of the scaled Schoenfeld residuals, along with a smooth curve using **ggplot2**. Wrapper around plot.cox.zph.

Usage

```
ggcoxzph(
  fit,
  resid = TRUE,
  se = TRUE,
  df = 4,
  nsmo = 40,
  var,
  point.col = "red",
  point.size = 1,
  point.shape = 19,
  point.alpha = 1,
  caption = NULL,
  ggtheme = theme_survminer(),
```

```
  ...
)

## S3 method for class 'ggcoxzph'
print(x, ..., newpage = TRUE)
```

Arguments

fit	an object of class cox.zph.
resid	a logical value, if TRUE the residuals are included on the plot, as well as the smooth fit.
se	a logical value, if TRUE, confidence bands at two standard errors will be added.
df	the degrees of freedom for the fitted natural spline, df=2 leads to a linear fit.
nsmo	number of points used to plot the fitted spline.
var	the set of variables for which plots are desired. By default, plots are produced in turn for each variable of a model.
point.col, point.size, point.shape, point.alpha	color, size, shape and visibility to be used for points.
caption	the caption of the final grob (bottom in arrangeGrob)
ggtheme	function, ggplot2 theme name. Allowed values include ggplot2 official themes: see theme.
...	further arguments passed to either the print() function or to the ggpar function for customizing the plot (see Details section).
x	an object of class ggcoxzph
newpage	open a new page. See grid.arrange.

Details

Customizing the plots: The plot can be easily customized using additional arguments to be passed to the function ggpar(). Read ?ggpubr::ggpar. These arguments include *font.main*,*font.submain*,*font.caption*,*font.x*,*font.y*,*font.* a vector of length 3 indicating respectively the size (e.g.: 14), the style (e.g.: "plain", "bold", "italic", "bold.italic") and the color (e.g.: "red") of main title, subtitle, caption, xlab and ylab and axis tick labels, respectively. For example *font.x = c(14, "bold", "red")*. Use font.x = 14, to change only font size; or use font.x = "bold", to change only font face.

Value

Returns an object of class ggcoxzph which is a list of ggplots.

Functions

- ggcoxzph: Graphical Test of Proportional Hazards using ggplot2.

Author(s)

Marcin Kosinski , <m.p.kosinski@gmail.com>

Examples

```
library(survival)
fit <- coxph(Surv(futime, fustat) ~ age + ecog.ps + rx, data=ovarian)
cox.zph.fit <- cox.zph(fit)
# plot all variables
ggcoxzph(cox.zph.fit)
# plot all variables in specified order
ggcoxzph(cox.zph.fit, var = c("ecog.ps", "rx", "age"), font.main = 12)
# plot specified variables in specified order
ggcoxzph(cox.zph.fit, var = c("ecog.ps", "rx"), font.main = 12, caption = "Caption goes here")
```

---

ggflexsurvplot *Ggplots of Fitted Flexible Survival Models*

Description

Create ggplot2-based graphs for flexible survival models.

Usage

```
ggflexsurvplot(
  fit,
  data = NULL,
  fun = c("survival", "cumhaz"),
  summary.flexsurv = NULL,
  size = 1,
  conf.int = FALSE,
  conf.int.flex = conf.int,
  conf.int.km = FALSE,
  legend.labs = NULL,
  ...
)
```

Arguments

`fit`	an object of class `flexsurvreg`.
`data`	the data used to fit survival curves.
`fun`	the type of survival curves. Allowed values include "survival" (default) and "cumhaz" (for cumulative hazard).
`summary.flexsurv`	(optional) the summary of the `flexsurvreg` object as generated by the function `summary()`.
`size`	line size for the flexible survival estimates.
`conf.int, conf.int.flex`	logical. If TRUE, add confidence bands for flexible survival estimates.

conf.int.km	same as conf.in.flex but for the kaplan-meier estimates.
legend.labs	character vector specifying legend labels. Used to replace the names of the strata from the fit. Should be given in the same order as those strata.
...	additional arguments passed to the function ggsurvplot().

Value

a ggsurvplot

Author(s)

Alboukadel Kassambara, <alboukadel.kassambara@gmail.com>

Examples

```
if(require("flexsurv")) {
fit <- flexsurvreg(Surv(rectime, censrec) ~ group,
                   dist = "gengamma", data = bc)
ggflexsurvplot(fit)
}
```

---

ggforest *Forest Plot for Cox Proportional Hazards Model*

Description

Drawing Forest Plot for Cox proportional hazards model. In two panels the model structure is presented.

Usage

```
ggforest(
  model,
  data = NULL,
  main = "Hazard ratio",
  cpositions = c(0.02, 0.22, 0.4),
  fontsize = 0.7,
  refLabel = "reference",
  noDigits = 2
)
```

Arguments

model	an object of class coxph.
data	a dataset used to fit survival curves. If not supplied then data will be extracted from 'fit' object.
main	title of the plot.
cpositions	relative positions of first three columns in the OX scale.
fontsize	relative size of annotations in the plot. Default value: 0.7.
refLabel	label for reference levels of factor variables.
noDigits	number of digits for estimates and p-values in the plot.

Value

returns a ggplot2 object (invisibly)

Author(s)

Przemyslaw Biecek (<przemyslaw.biecek@gmail.com>), Fabian Scheipl (<fabian.scheipl@gmail.com>)

Examples

```
require("survival")
model <- coxph( Surv(time, status) ~ sex + rx + adhere,
                data = colon )
ggforest(model)

colon <- within(colon, {
  sex <- factor(sex, labels = c("female", "male"))
  differ <- factor(differ, labels = c("well", "moderate", "poor"))
  extent <- factor(extent, labels = c("submuc.", "muscle", "serosa", "contig."))
})
bigmodel <-
  coxph(Surv(time, status) ~ sex + rx + adhere + differ + extent + node4,
    data = colon )
ggforest(bigmodel)
```

---

ggrisktable *Plot Survival Tables*

Description

Plot survival tables:

- ggrisktable(): Plot the number at risk table.
- ggcumevents(): Plot the cumulative number of events table.

- ggcumcensor(): Plot the cumulative number of censored subjects, the number of subjects who exit the risk set, without an event, at time t. Normally, users don't need to use this function directly.
- ggsurvtable(): Generic function to plot any survival tables.

Normally, users don't need to use this function directly. Internally used by the function ggsurvplot.

Usage

```
ggrisktable(
  fit,
  data = NULL,
  risk.table.type = c("absolute", "percentage", "abs_pct", "nrisk_cumcensor",
    "nrisk_cumevents"),
  ...
)

ggcumevents(fit, data = NULL, ...)

ggcumcensor(fit, data = NULL, ...)

ggsurvtable(
  fit,
  data = NULL,
  survtable = c("cumevents", "cumcensor", "risk.table"),
  risk.table.type = c("absolute", "percentage", "abs_pct", "nrisk_cumcensor",
    "nrisk_cumevents"),
  title = NULL,
  risk.table.title = NULL,
  cumevents.title = title,
  cumcensor.title = title,
  color = "black",
  palette = NULL,
  break.time.by = NULL,
  xlim = NULL,
  xscale = 1,
  xlab = "Time",
  ylab = "Strata",
  xlog = FALSE,
  legend = "top",
  legend.title = "Strata",
  legend.labs = NULL,
  y.text = TRUE,
  y.text.col = TRUE,
  fontsize = 4.5,
  font.family = "",
  axes.offset = TRUE,
  ggtheme = theme_survminer(),
  tables.theme = ggtheme,
```

```
  ...
)
```

Arguments

fit	an object of class survfit. Can be a list containing two components: 1) time: time variable used in survfit; 2) table: survival table as generated by the internal function .get_timepoints_survsummary(). Can be also a simple data frame.
data	a dataset used to fit survival curves. If not supplied then data will be extracted from 'fit' object.
risk.table.type	risk table type. Allowed values include: "absolute" or "percentage": to show the **absolute number** and the **percentage** of subjects at risk by time, respectively. Use "abs_pct" to show both absolute number and percentage. Used only when survtable = "risk.table".
...	other arguments passed to the function ggsurvtable and ggpar.
survtable	a character string specifying the type of survival table to plot.
title	the title of the plot.
risk.table.title	The title to be used for the risk table.
cumevents.title	The title to be used for the cumulative events table.
cumcensor.title	The title to be used for the cumcensor table.
color	color to be used for the survival curves. • If the number of strata/group (n.strata) = 1, the expected value is the color name. For example color = "blue". • If n.strata > 1, the expected value is the grouping variable name. By default, survival curves are colored by strata using the argument color = "strata", but you can also color survival curves by any other grouping variables used to fit the survival curves. In this case, it's possible to specify a custom color palette by using the argument palette.
palette	the color palette to be used. Allowed values include "hue" for the default hue color scale; "grey" for grey color palettes; brewer palettes e.g. "RdBu", "Blues", ...; or custom color palette e.g. c("blue", "red"); and scientific journal palettes from ggsci R package, e.g.: "npg", "aaas", "lancet", "jco", "ucscgb", "uchicago", "simpsons" and "rickandmorty". See details section for more information. Can be also a numeric vector of length(groups); in this case a basic color palette is created using the function palette.
break.time.by	numeric value controlling time axis breaks. Default value is NULL.
xlim	x and y axis limits e.g. xlim = c(0, 1000), ylim = c(0, 1).
xscale	numeric or character value specifying x-axis scale. • If numeric, the value is used to divide the labels on the x axis. For example, a value of 365.25 will give labels in years instead of the original days.

- If character, allowed options include one of c("d_m", "d_y", "m_d", "m_y", "y_d", "y_m"), where d = days, m = months and y = years. For example, xscale = "d_m" will transform labels from days to months; xscale = "m_y", will transform labels from months to years.

`xlab`	main title and axis labels
`ylab`	main title and axis labels
`xlog`	logical value. If TRUE, x axis is tansformed into log scale.
`legend`	character specifying legend position. Allowed values are one of c("top", "bottom", "left", "right", "none"). Default is "top" side position. to remove the legend use legend = "none". Legend position can be also specified using a numeric vector c(x, y); see details section.
`legend.title`	legend title.
`legend.labs`	character vector specifying legend labels. Used to replace the names of the strata from the fit. Should be given in the same order as those strata.
`y.text`	logical. Default is TRUE. If FALSE, the table y axis tick labels will be hidden.
`y.text.col`	logical. Default value is FALSE. If TRUE, the table tick labels will be colored by strata.
`fontsize`	text font size.
`font.family`	character vector specifying text element font family, e.g.: font.family = "Courier New".
`axes.offset`	logical value. Default is TRUE. If FALSE, set the plot axes to start at the origin.
`ggtheme`	function, ggplot2 theme name. Default value is theme_survminer. Allowed values include ggplot2 official themes: see `theme`.
`tables.theme`	function, ggplot2 theme name. Default value is theme_survminer. Allowed values include ggplot2 official themes: see `theme`. Note that, `tables.theme` is incremental to `ggtheme`.

Value

a ggplot.

Functions

- `ggrisktable`: Plot the number at risk table.
- `ggcumevents`: Plot the cumulative number of events table
- `ggcumcensor`: Plot the cumulative number of censor table
- `ggsurvtable`: Generic function to plot survival tables: risk.table, cumevents and cumcensor

Author(s)

Alboukadel Kassambara, <alboukadel.kassambara@gmail.com>

Examples

```
# Fit survival curves
#:::::::::::::::::::::::::::::::::::::::::::::
require("survival")
fit<- survfit(Surv(time, status) ~ sex, data = lung)

# Survival tables
#:::::::::::::::::::::::::::::::::::::::::::::
tables <- ggsurvtable(fit, data = lung, color = "strata",
  y.text = FALSE)

# Risk table
tables$risk.table

# Number of cumulative events
tables$cumevents

# Number of cumulative censoring
tables$cumcensor
```

---

ggsurvevents *Distribution of Events' Times*

Description

Distribution of Events' Times

Usage

```
ggsurvevents(
  surv = NULL,
  fit = NULL,
  data = NULL,
  type = "fraction",
  normalized = TRUE,
  censored.on.top = TRUE,
  ggtheme = theme_survminer(),
  palette = c("grey75", "grey25"),
  ...
)
```

Arguments

surv	an object of Surv. If not suplied, the censoring variable is extracted from the model.
fit	an object of class survfit.
data	a dataset for predictions. If not supplied then data will be extracted from 'fit' object.

type	one of c("cumulative","radius","fraction"). "cumulative" stands for cumulative number of events, "radius" stands for number of events within a given radius,
normalized	if TRUE relative number of events is presented,
censored.on.top	is TRUE then censored events are on the top
ggtheme	function, ggplot2 theme name. Allowed values include ggplot2 official themes: see theme.
palette	the color palette to be used for coloring of significant variables.
...	other graphical parameters to be passed to the function ggpar.

Value

return an object of class ggplot

Author(s)

Przemyslaw Biecek, <przemyslaw.biecek@gmail.com>

Examples

```
require("survival")
# from Surv
surv <- Surv(lung$time, lung$status)
ggsurvevents(surv)

surv2 <- Surv(colon$time, colon$status)
ggsurvevents(surv2)
ggsurvevents(surv2, normalized = TRUE)

# from survfit
fit <- survfit(Surv(time, status) ~ sex, data = lung)
ggsurvevents(fit = fit, data = lung)

# from coxph
model <- coxph( Surv(time, status) ~ sex + rx + adhere, data = colon )
ggsurvevents(fit = model, data = colon)
ggsurvevents(surv2, normalized = TRUE, type = "radius")
ggsurvevents(surv2, normalized = TRUE, type = "fraction")
```

---

ggsurvplot *Drawing Survival Curves Using ggplot2*

Description

ggsurvplot() is a generic function to plot survival curves. Wrapper around the ggsurvplot_xx() family functions. Plot one or a list of survfit objects as generated by the survfit.formula() and surv_fit functions:

- ggsurvplot_list()
- ggsurvplot_facet()
- ggsurvplot_group_by()
- ggsurvplot_add_all()
- ggsurvplot_combine()

See the documentation for each function to learn how to control that aspect of the ggsurvplot(). ggsurvplot() accepts further arguments to be passed to the ggsurvplot_xx() functions. Has options to:

- plot a list of survfit objects,
- facet survival curves into multiple panels,
- group dataset by one or two grouping variables and to create the survival curves in each subset,
- combine multiple survfit objects into one plot,
- add survival curves of the pooled patients (null model) onto the main stratified plot,
- plot survival curves from a data frame containing survival curve summary as returned by surv_summary().

Usage

```
ggsurvplot(
  fit,
  data = NULL,
  fun = NULL,
  color = NULL,
  palette = NULL,
  linetype = 1,
  conf.int = FALSE,
  pval = FALSE,
  pval.method = FALSE,
  test.for.trend = FALSE,
  surv.median.line = "none",
  risk.table = FALSE,
  cumevents = FALSE,
  cumcensor = FALSE,
  tables.height = 0.25,
  group.by = NULL,
  facet.by = NULL,
  add.all = FALSE,
  combine = FALSE,
  ggtheme = theme_survminer(),
```

```
  tables.theme = ggtheme,
  ...
)

## S3 method for class 'ggsurvplot'
print(
  x,
  surv.plot.height = NULL,
  risk.table.height = NULL,
  ncensor.plot.height = NULL,
  newpage = TRUE,
  ...
)
```

Arguments

fit	allowed values include: • a survfit object • a list of survfit objects. Passed to ggsurvplot_list() • a data frame containing survival curves summary. Passed to ggsurvplot_df().
data	a dataset used to fit survival curves. If not supplied then data will be extracted from 'fit' object.
fun	an arbitrary function defining a transformation of the survival curve. Often used transformations can be specified with a character argument: "event" plots cumulative events (f(y) = 1-y), "cumhaz" plots the cumulative hazard function (f(y) = -log(y)), and "pct" for survival probability in percentage.
color	color to be used for the survival curves. • If the number of strata/group (n.strata) = 1, the expected value is the color name. For example color = "blue". • If n.strata > 1, the expected value is the grouping variable name. By default, survival curves are colored by strata using the argument color = "strata", but you can also color survival curves by any other grouping variables used to fit the survival curves. In this case, it's possible to specify a custom color palette by using the argument palette.
palette	the color palette to be used. Allowed values include "hue" for the default hue color scale; "grey" for grey color palettes; brewer palettes e.g. "RdBu", "Blues", ...; or custom color palette e.g. c("blue", "red"); and scientific journal palettes from ggsci R package, e.g.: "npg", "aaas", "lancet", "jco", "ucscgb", "uchicago", "simpsons" and "rickandmorty". See details section for more information. Can be also a numeric vector of length(groups); in this case a basic color palette is created using the function palette.
linetype	line types. Allowed values includes i) "strata" for changing linetypes by strata (i.e. groups); ii) a numeric vector (e.g., c(1, 2)) or a character vector c("solid", "dashed").
conf.int	logical value. If TRUE, plots confidence interval.

`pval` logical value, a numeric or a string. If logical and TRUE, the p-value is added on the plot. If numeric, than the computet p-value is substituted with the one passed with this parameter. If character, then the customized string appears on the plot. See examples - Example 3.

`pval.method` whether to add a text with the test name used for calculating the pvalue, that corresponds to survival curves' comparison - used only when `pval=TRUE`

`test.for.trend` logical value. Default is FALSE. If TRUE, returns the test for trend p-values. Tests for trend are designed to detect ordered differences in survival curves. That is, for at least one group. The test for trend can be only performed when the number of groups is > 2.

`surv.median.line`
character vector for drawing a horizontal/vertical line at median survival. Allowed values include one of c("none", "hv", "h", "v"). v: vertical, h:horizontal.

`risk.table` Allowed values include:

- TRUE or FALSE specifying whether to show or not the risk table. Default is FALSE.
- "absolute" or "percentage". Shows the **absolute number** and the **percentage** of subjects at risk by time, respectively.
- "abs_pct" to show both absolute number and percentage.
- "nrisk_cumcensor" and "nrisk_cumevents". Show the number at risk and, the cumulative number of censoring and events, respectively.

`cumevents` logical value specifying whether to show or not the table of the cumulative number of events. Default is FALSE.

`cumcensor` logical value specifying whether to show or not the table of the cumulative number of censoring. Default is FALSE.

`tables.height` numeric value (in [0 - 1]) specifying the general height of all tables under the main survival plot.

`group.by` a character vector containing the name of grouping variables. Should be of length <= 2. Alias of the `ggsurvplot_group_by()` function.

`facet.by` a character vector containing the name of grouping variables to facet the survival curves into multiple panels. Should be of length <= 2. Alias of the `ggsurvplot_facet()` function.

`add.all` a logical value. If TRUE, add the survival curve of pooled patients (null model) onto the main plot. Alias of the `ggsurvplot_add_all()` function.

`combine` a logical value. If TRUE, combine a list survfit objects on the same plot. Alias of the `ggsurvplot_combine()` function.

`ggtheme` function, ggplot2 theme name. Default value is theme_survminer. Allowed values include ggplot2 official themes: see `theme`.

`tables.theme` function, ggplot2 theme name. Default value is theme_survminer. Allowed values include ggplot2 official themes: see `theme`. Note that, `tables.theme` is incremental to `ggtheme`.

`...` Futher arguments as described hereafter and other arguments to be passed i) to ggplot2 geom_*() functions such as linetype, size, ii) or to the function ggpar() for customizing the plots. See details section.

x	an object of class ggsurvplot
surv.plot.height	the height of the survival plot on the grid. Default is 0.75. Ignored when risk.table = FALSE.
risk.table.height	the height of the risk table on the grid. Increase the value when you have many strata. Default is 0.25. Ignored when risk.table = FALSE.
ncensor.plot.height	The height of the censor plot. Used when ncensor.plot = TRUE.
newpage	open a new page. See grid.arrange

Details

- **Color palettes**: The argument **palette** can be used to specify the color to be used for each group. By default, the first color in the palette is used to color the first level of the factor variable. This default behavior can be changed by assigning correctly a named vector. That is, the names of colors should match the strata names as generated by the ggsurvplot() function in the legend.

Value

return an object of class ggsurvplot which is list containing the following components:

- plot: the survival plot (ggplot object)
- table: the number of subjects at risk table per time (ggplot object).
- cumevents: the cumulative number of events table (ggplot object).
- ncensor.plot: the number of censoring (ggplot object).
- data.survplot: the data used to plot the survival curves (data.frame).
- data.survtable: the data used to plot the tables under the main survival curves (data.frame).

FURTHER ARGUMENTS

Customize survival plots and tables. See also ggsurvplot_arguments.

Plot title and axis labels

- **title**: main title.
- **xlab, ylab**: x and y axis labels, respectively.

Legend title, labels and position

- **legend**: character specifying legend position. Allowed values are one of c("top", "bottom", "left", "right", "none"). Default is "top" side position. to remove the legend use legend = "none". Legend position can be also specified using a numeric vector c(x, y). In this case it is possible to position the legend inside the plotting area. x and y are the coordinates of the legend box. Their values should be between 0 and 1. c(0,0) corresponds to the "bottom left" and c(1,1) corresponds to the "top right" position. For instance use legend = c(0.8, 0.2).

- **legend.title**: legend title.
- **legend.labs**: character vector specifying legend labels. Used to replace the names of the strata from the fit. Should be given in the same order as those strata.

Axis limits, breaks and scales

- **break.time.by**: numeric value controlling time axis breaks. Default value is NULL.
- **break.x.by**: alias of break.time.by. Numeric value controlling x axis breaks. Default value is NULL.
- **break.y.by**: same as break.x.by but for y axis.
- **surv.scale**: scale transformation of survival curves. Allowed values are "default" or "percent".
- **xscale**: numeric or character value specifying x-axis scale.
  - If numeric, the value is used to divide the labels on the x axis. For example, a value of 365.25 will give labels in years instead of the original days.
  - If character, allowed options include one of - "d_m", "d_y", "m_d", "m_y", "y_d" and "y_m" - where `d` = `days`, `m` = `months` and `y` = `years`. For example, `xscale` = `"d_m"` will transform labels from days to months; `xscale` = `"m_y"`, will transform labels from months to years.
- **xlim,ylim**: x and y axis limits e.g. xlim = c(0, 1000), ylim = c(0, 1).
- **axes.offset**: logical value. Default is TRUE. If FALSE, set the plot axes to start at the origin.

Confidence interval

- **conf.int.fill**: fill color to be used for confidence interval.
- **conf.int.style**: confidence interval style. Allowed values include c("ribbon", "step").
- **conf.int.alpha**: numeric value specifying confidence fill color transparency. Value should be in [0, 1], where 0 is full transparency and 1 is no transparency.

P-value

- **pval.size**: numeric value specifying the p-value text size. Default is 5.
- **pval.coord**: numeric vector, of length 2, specifying the x and y coordinates of the p-value. Default values are NULL.
- **pval.method.size**: the same as `pval.size` but for displaying `log.rank.weights` name.
- **pval.method.coord**: the same as `pval.coord` but for displaying `log.rank.weights` name.
- **log.rank.weights**: the name for the type of weights to be used in computing the p-value for log-rank test. By default `survdiff` is used to calculate regular log-rank test (with weights == 1). A user can specify `"1"`,`"n"`,`"sqrtN"`,`"S1"`,`"S2"`,`"FH"` to use weights specified in comp, so that weight correspond to the test as : 1 - log-rank, n - Gehan-Breslow (generalized Wilcoxon), sqrtN - Tarone-Ware, S1 - Peto-Peto's modified survival estimate, S2 - modified Peto-Peto (by Andersen), FH - Fleming-Harrington(p=1, q=1).

Median survival

- **surv.median.line**: character vector for drawing a horizontal/vertical line at median survival. Allowed values include one of c("none", "hv", "h", "v"). v: vertical, h:horizontal.

Censor points

- **censor**: logical value. If TRUE (default), censors will be drawn.
- **censor.shape**: character or numeric value specifying the point shape of censors. Default value is "+" (3), a sensible choice is "|" (124).
- **censor.size**: numveric value specifying the point size of censors. Default is 4.5.

Survival tables

General parameters for all tables. The arguments below, when specified, will be applied to all survival tables at once (risk, cumulative events and cumulative censoring tables).

- **tables.col**: color to be used for all tables under the main plot. Default value is "black". If you want to color by strata (i.e. groups), use tables.col = "strata".
- **fontsize**: font size to be used for the risk table and the cumulative events table.
- **font.family**: character vector specifying text element font family, e.g.: font.family = "Courier New".
- **tables.y.text**: logical. Default is TRUE. If FALSE, the y axis tick labels of tables will be hidden.
- **tables.y.text.col**: logical. Default value is FALSE. If TRUE, the y tick labels of tables will be colored by strata.
- **tables.height**: numeric value (in [0 - 1]) specifying the general height of all tables under the main survival plot. Increase the value when you have many strata. Default is 0.25.

Specific to the risk table

- **risk.table.title**: the title to be used for the risk table.
- **risk.table.pos**: character vector specifying the risk table position. Allowed options are one of c("out", "in") indicating 'outside' or 'inside' the main plot, respectively. Default value is "out".
- `risk.table.col`, `risk.table.fontsize`, `risk.table.y.text`, `risk.table.y.text.col` and `risk.table.height`: same as for the general parameters but applied to the risk table only.

Specific to the number of cumulative events table (cumevents)

- **cumevents.title**: the title to be used for the cumulative events table.
- `cumevents.col,cumevents.y.text,cumevents.y.text,cumevents.height`: same as for the general parameters but for the cumevents table only.

Specific to the number of cumulative censoring table (cumcensor)

- **cumcensor.title**: the title to be used for the cumcensor table.
- `cumcensor.col`, `cumcensor.y.text`, `cumcensor.y.text.col`, `cumcensor.height`: same as for the general parameters but for cumcensor table only.

Survival plot height

- **surv.plot.height**: the height of the survival plot on the grid. Default is 0.75. Ignored when risk.table = FALSE.

Number of censored subjects barplot

- **ncensor.plot**: logical value. If TRUE, the number of censored subjects at time t is plotted. Default is FALSE. Ignored when cumcensor = TRUE.
- **ncensor.plot.title**: the title to be used for the censor plot. Used when `ncensor.plot = TRUE`.
- **ncensor.plot.height**: the height of the censor plot. Used when `ncensor.plot = TRUE`.

Other graphical parameters

The plot can be easily customized using additional arguments to be passed to the function `ggpar()`.

These arguments include `font.title,font.subtitle,font.caption,font.x,font.y,font.tickslab` and `font.legend`, which are vectors of length 3 indicating respectively the size (e.g.: 14), the style (e.g.: "plain", "bold", "italic", "bold.italic") and the color (e.g.: "red") of main title, subtitle, caption, xlab and ylab, axis tick labels and legend, respectively. For example font.x = c(14, "bold", "red").

Use font.x = 14, to change only font size; or use font.x = "bold", to change only font face.

Author(s)

Alboukadel Kassambara, <alboukadel.kassambara@gmail.com>

Examples

```
#%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
# Example 1: Survival curves with two groups
#%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

# Fit survival curves
#++++++++++++++++++++++++++++++++++++
require("survival")
fit<- survfit(Surv(time, status) ~ sex, data = lung)

# Basic survival curves
ggsurvplot(fit, data = lung)

# Customized survival curves
ggsurvplot(fit, data = lung,
 surv.median.line = "hv", # Add medians survival

 # Change legends: title & labels
 legend.title = "Sex",
 legend.labs = c("Male", "Female"),
 # Add p-value and tervals
 pval = TRUE,

 conf.int = TRUE,
 # Add risk table
 risk.table = TRUE,
 tables.height = 0.2,
 tables.theme = theme_cleantable(),
```

```
 # Color palettes. Use custom color: c("#E7B800", "#2E9FDF"),
 # or brewer color (e.g.: "Dark2"), or ggsci color (e.g.: "jco")
 palette = c("#E7B800", "#2E9FDF"),
 ggtheme = theme_bw() # Change ggplot2 theme
)

# Change font size, style and color
#++++++++++++++++++++++++++++++++++
## Not run:
# Change font size, style and color at the same time
ggsurvplot(fit, data = lung,  main = "Survival curve",
   font.main = c(16, "bold", "darkblue"),
   font.x = c(14, "bold.italic", "red"),
   font.y = c(14, "bold.italic", "darkred"),
   font.tickslab = c(12, "plain", "darkgreen"))

## End(Not run)



#%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
# Example 2: Facet ggsurvplot() output by
# a combination of factors
#%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

# Fit (complexe) survival curves
#++++++++++++++++++++++++++++++++++
## Not run:
require("survival")
fit3 <- survfit( Surv(time, status) ~ sex + rx + adhere,
                data = colon )

# Visualize
#++++++++++++++++++++++++++++++++++
ggsurv <- ggsurvplot(fit3, data = colon,
  fun = "cumhaz", conf.int = TRUE,
  risk.table = TRUE, risk.table.col="strata",
  ggtheme = theme_bw())

# Faceting survival curves
curv_facet <- ggsurv$plot + facet_grid(rx ~ adhere)
curv_facet

# Faceting risk tables:
# Generate risk table for each facet plot item
ggsurv$table + facet_grid(rx ~ adhere, scales = "free")+
 theme(legend.position = "none")

 # Generate risk table for each facet columns
tbl_facet <- ggsurv$table + facet_grid(.~ adhere, scales = "free")
tbl_facet + theme(legend.position = "none")

# Arrange faceted survival curves and risk tables
```

```
g2 <- ggplotGrob(curv_facet)
g3 <- ggplotGrob(tbl_facet)
min_ncol <- min(ncol(g2), ncol(g3))
g <- gridExtra::gtable_rbind(g2[, 1:min_ncol], g3[, 1:min_ncol], size="last")
g$widths <- grid::unit.pmax(g2$widths, g3$widths)
grid::grid.newpage()
grid::grid.draw(g)


## End(Not run)

#%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
# Example 3: CUSTOMIZED PVALUE
#%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
# Customized p-value
ggsurvplot(fit, data = lung, pval = TRUE)
ggsurvplot(fit, data = lung, pval = 0.03)
ggsurvplot(fit, data = lung, pval = "The hot p-value is: 0.031")
```

`ggsurvplot_add_all` *Add Survival Curves of Pooled Patients onto the Main Plot*

Description

Add survival curves of pooled patients onto the main plot stratified by grouping variables.

Usage

```
ggsurvplot_add_all(
  fit,
  data,
  legend.title = "Strata",
  legend.labs = NULL,
  pval = FALSE,
  ...
)
```

Arguments

`fit`	an object of class survfit.
`data`	a dataset used to fit survival curves. If not supplied then data will be extracted from 'fit' object.
`legend.title`	legend title.
`legend.labs`	character vector specifying legend labels. Used to replace the names of the strata from the fit. Should be given in the same order as those strata.

pval	logical value, a numeric or a string. If logical and TRUE, the p-value is added on the plot. If numeric, than the computet p-value is substituted with the one passed with this parameter. If character, then the customized string appears on the plot. See examples - Example 3.
...	other arguments passed to the ggsurvplot() function.

Value

Return a ggsurvplot.

See Also

ggsurvplot

Examples

```
library(survival)

# Fit survival curves
fit <- surv_fit(Surv(time, status) ~ sex, data = lung)

# Visualize survival curves
ggsurvplot(fit, data = lung,
          risk.table = TRUE, pval = TRUE,
          surv.median.line = "hv", palette = "jco")

# Add survival curves of pooled patients (Null model)
# Use add.all = TRUE option
ggsurvplot(fit, data = lung,
          risk.table = TRUE, pval = TRUE,
          surv.median.line = "hv", palette = "jco",
          add.all = TRUE)
```

---

ggsurvplot_arguments *ggsurvplot Argument Descriptions*

Description

ggsurvplot Argument Descriptions

Arguments

fit	an object of class survfit.
data	a dataset used to fit survival curves. If not supplied then data will be extracted from 'fit' object.

`fun`	an arbitrary function defining a transformation of the survival curve. Often used transformations can be specified with a character argument: "event" plots cumulative events (f(y) = 1-y), "cumhaz" plots the cumulative hazard function (f(y) = -log(y)), and "pct" for survival probability in percentage.
`surv.scale`	scale transformation of survival curves. Allowed values are "default" or "percent".
`xscale`	numeric or character value specifying x-axis scale. • If numeric, the value is used to divide the labels on the x axis. For example, a value of 365.25 will give labels in years instead of the original days. • If character, allowed options include one of c("d_m", "d_y", "m_d", "m_y", "y_d", "y_m"), where d = days, m = months and y = years. For example, xscale = "d_m" will transform labels from days to months; xscale = "m_y", will transform labels from months to years.
`color`	color to be used for the survival curves. • If the number of strata/group (n.strata) = 1, the expected value is the color name. For example color = "blue". • If n.strata > 1, the expected value is the grouping variable name. By default, survival curves are colored by strata using the argument color = "strata", but you can also color survival curves by any other grouping variables used to fit the survival curves. In this case, it's possible to specify a custom color palette by using the argument palette.
`palette`	the color palette to be used. Allowed values include "hue" for the default hue color scale; "grey" for grey color palettes; brewer palettes e.g. "RdBu", "Blues", ...; or custom color palette e.g. c("blue", "red"); and scientific journal palettes from ggsci R package, e.g.: "npg", "aaas", "lancet", "jco", "ucscgb", "uchicago", "simpsons" and "rickandmorty". See details section for more information. Can be also a numeric vector of length(groups); in this case a basic color palette is created using the function palette.
`linetype`	line types. Allowed values includes i) "strata" for changing linetypes by strata (i.e. groups); ii) a numeric vector (e.g., c(1, 2)) or a character vector c("solid", "dashed").
`break.time.by`	numeric value controlling time axis breaks. Default value is NULL.
`break.x.by`	alias of break.time.by. Numeric value controlling x axis breaks. Default value is NULL.
`break.y.by`	same as break.x.by but for y axis.
`conf.int`	logical value. If TRUE, plots confidence interval.
`conf.int.fill`	fill color to be used for confidence interval.
`conf.int.style`	confidence interval style. Allowed values include c("ribbon", "step").
`conf.int.alpha`	numeric value specifying fill color transparency. Value should be in [0, 1], where 0 is full transparency and 1 is no transparency.
`censor`	logical value. If TRUE, censors will be drawn.
`censor.shape`	character or numeric value specifying the point shape of censors. Default value is "+" (3), a sensible choice is "|" (124).

`censor.size` numveric value specifying the point size of censors. Default is 4.5.

`pval` logical value, a numeric or a string. If logical and TRUE, the p-value is added on the plot. If numeric, than the computet p-value is substituted with the one passed with this parameter. If character, then the customized string appears on the plot. See examples - Example 3.

`pval.size` numeric value specifying the p-value text size. Default is 5.

`pval.coord` numeric vector, of length 2, specifying the x and y coordinates of the p-value. Default values are NULL.

`title, xlab, ylab` main title and axis labels

`xlim, ylim` x and y axis limits e.g. xlim = c(0, 1000), ylim = c(0, 1).

`axes.offset` logical value. Default is TRUE. If FALSE, set the plot axes to start at the origin.

`legend` character specifying legend position. Allowed values are one of c("top", "bottom", "left", "right", "none"). Default is "top" side position. to remove the legend use legend = "none". Legend position can be also specified using a numeric vector c(x, y); see details section.

`legend.title` legend title.

`legend.labs` character vector specifying legend labels. Used to replace the names of the strata from the fit. Should be given in the same order as those strata.

`risk.table` Allowed values include:

- TRUE or FALSE specifying whether to show or not the risk table. Default is FALSE.
- "absolute" or "percentage". Shows the **absolute number** and the **percentage** of subjects at risk by time, respectively.
- "abs_pct" to show both absolute number and percentage.
- "nrisk_cumcensor" and "nrisk_cumevents". Show the number at risk and, the cumulative number of censoring and events, respectively.

`risk.table.title` The title to be used for the risk table.

`risk.table.pos` character vector specifying the risk table position. Allowed options are one of c("out", "in") indicating 'outside' or 'inside' the main plot, respectively. Default value is "out".

`risk.table.col` same as tables.col but for risk table only.

`risk.table.fontsize, fontsize` font size to be used for the risk table and the cumulative events table.

`risk.table.y.text` logical. Default is TRUE. If FALSE, risk table y axis tick labels will be hidden.

`risk.table.y.text.col` logical. Default value is FALSE. If TRUE, risk table tick labels will be colored by strata.

`tables.height` numeric value (in [0 - 1]) specifying the general height of all tables under the main survival plot.

`tables.y.text` logical. Default is TRUE. If FALSE, the y axis tick labels of tables will be hidden.

`tables.y.text.col`
: logical. Default value is FALSE. If TRUE, tables tick labels will be colored by strata.

`tables.col`
: color to be used for all tables under the main plot. Default value is "black". If you want to color by strata (i.e. groups), use tables.col = "strata".

`tables.theme`
: function, ggplot2 theme name. Default value is theme_survminer. Allowed values include ggplot2 official themes: see `theme`. Note that, `tables.theme` is incremental to `ggtheme`.

`risk.table.height`
: the height of the risk table on the grid. Increase the value when you have many strata. Default is 0.25. Ignored when risk.table = FALSE.

`surv.plot.height`
: the height of the survival plot on the grid. Default is 0.75. Ignored when risk.table = FALSE.

`ncensor.plot`
: logical value. If TRUE, the number of censored subjects at time t is plotted. Default is FALSE. Ignored when cumcensor = TRUE.

`ncensor.plot.title`
: The title to be used for the censor plot. Used when `ncensor.plot` = `TRUE`.

`ncensor.plot.height`
: The height of the censor plot. Used when `ncensor.plot` = `TRUE`.

`cumevents`
: logical value specifying whether to show or not the table of the cumulative number of events. Default is FALSE.

`cumevents.title`
: The title to be used for the cumulative events table.

`cumevents.col`
: same as tables.col but for the cumulative events table only.

`cumevents.y.text`
: logical. Default is TRUE. If FALSE, the y axis tick labels of the cumulative events table will be hidden.

`cumevents.y.text.col`
: logical. Default value is FALSE. If TRUE, the y tick labels of the cumulative events will be colored by strata.

`cumevents.height`
: the height of the cumulative events table on the grid. Default is 0.25. Ignored when cumevents = FALSE.

`cumcensor`
: logical value specifying whether to show or not the table of the cumulative number of censoring. Default is FALSE.

`cumcensor.title`
: The title to be used for the cumcensor table.

`cumcensor.col`
: same as tables.col but for cumcensor table only.

`cumcensor.y.text`
: logical. Default is TRUE. If FALSE, the y axis tick labels of the cumcensor table will be hidden.

`cumcensor.y.text.col`
: logical. Default value is FALSE. If TRUE, the y tick labels of the cumcensor will be colored by strata.

cumcensor.height
: the height of the cumcensor table on the grid. Default is 0.25. Ignored when cumcensor = FALSE.

surv.median.line
: character vector for drawing a horizontal/vertical line at median survival. Allowed values include one of c("none", "hv", "h", "v"). v: vertical, h:horizontal.

ggtheme
: function, ggplot2 theme name. Default value is theme_survminer. Allowed values include ggplot2 official themes: see theme.

...
: other arguments to be passed i) to ggplot2 geom_*() functions such as linetype, size, ii) or to the function ggpar() for customizing the plots. See details section.

log.rank.weights
: The name for the type of weights to be used in computing the p-value for log-rank test. By default survdiff is used to calculate regular log-rank test (with weights == 1). A user can specify "1","n","sqrtN","S1","S2","FH" to use weights specified in comp, so that weight correspond to the test as : 1 - log-rank, n - Gehan-Breslow (generalized Wilcoxon), sqrtN - Tarone-Ware, S1 - Peto-Peto's modified survival estimate, S2 - modified Peto-Peto (by Andersen), FH - Fleming-Harrington(p=1, q=1).

pval.method
: whether to add a text with the test name used for calculating the pvalue, that corresponds to survival curves' comparison - used only when pval=TRUE

pval.method.size
: the same as pval.size but for displaying log.rank.weights name

pval.method.coord
: the same as pval.coord but for displaying log.rank.weights name

---

ggsurvplot_combine *Combine a List of Survfit Objects on the Same Plot*

Description

Combine multiple survfit objects on the same plot. For example, one might wish to plot progression free survival and overall survival on the same graph (and also stratified by treatment assignment). ggsurvplot_combine() provides an extension to the ggsurvplot() function for doing that.

Usage

```
ggsurvplot_combine(
  fit,
  data,
  risk.table = FALSE,
  risk.table.pos = c("out", "in"),
  cumevents = FALSE,
  cumcensor = FALSE,
  tables.col = "black",
  tables.y.text = TRUE,
  tables.y.text.col = TRUE,
```

```
  ggtheme = theme_survminer(),
  tables.theme = ggtheme,
  keep.data = FALSE,
  risk.table.y.text = tables.y.text,
  ...
)
```

Arguments

fit	a named list of survfit objects.
data	the data frame used to compute survival curves.
risk.table	Allowed values include: • TRUE or FALSE specifying whether to show or not the risk table. Default is FALSE. • "absolute" or "percentage". Shows the **absolute number** and the **percentage** of subjects at risk by time, respectively. • "abs_pct" to show both absolute number and percentage. • "nrisk_cumcensor" and "nrisk_cumevents". Show the number at risk and, the cumulative number of censoring and events, respectively.
risk.table.pos	character vector specifying the risk table position. Allowed options are one of c("out", "in") indicating 'outside' or 'inside' the main plot, respectively. Default value is "out".
cumevents	logical value specifying whether to show or not the table of the cumulative number of events. Default is FALSE.
cumcensor	logical value specifying whether to show or not the table of the cumulative number of censoring. Default is FALSE.
tables.col	color to be used for all tables under the main plot. Default value is "black". If you want to color by strata (i.e. groups), use tables.col = "strata".
tables.y.text	logical. Default is TRUE. If FALSE, the y axis tick labels of tables will be hidden.
tables.y.text.col	logical. Default value is FALSE. If TRUE, tables tick labels will be colored by strata.
ggtheme	function, ggplot2 theme name. Default value is theme_survminer. Allowed values include ggplot2 official themes: see theme.
tables.theme	function, ggplot2 theme name. Default value is theme_survminer. Allowed values include ggplot2 official themes: see theme. Note that, tables.theme is incremental to ggtheme.
keep.data	logical value specifying whether the plot data frame should be kept in the result. Setting these to FALSE (default) can give much smaller results and hence even save memory allocation time.
risk.table.y.text	logical. Default is TRUE. If FALSE, risk table y axis tick labels will be hidden.
...	other arguments to pass to the ggsurvplot() function.

Examples

```
library(survival)
# Create a demo data set
#:::::::::::::::::::::::::::::::::::::::::::::::::::::::
 set.seed(123)
 demo.data <- data.frame(
   os.time = colon$time,
   os.status = colon$status,
   pfs.time = sample(colon$time),
   pfs.status = colon$status,
   sex = colon$sex, rx = colon$rx, adhere = colon$adhere
 )

# Ex1: Combine null models
#:::::::::::::::::::::::::::::::::::::::::::::::::::::::
 # Fit
 pfs <- survfit( Surv(pfs.time, pfs.status) ~ 1, data = demo.data)
 os <- survfit( Surv(os.time, os.status) ~ 1, data = demo.data)
 # Combine on the same plot
 fit <- list(PFS = pfs, OS = os)
 ggsurvplot_combine(fit, demo.data)

# Combine survival curves stratified by treatment assignment rx
#:::::::::::::::::::::::::::::::::::::::::::::::::::::::
# Fit
pfs <- survfit( Surv(pfs.time, pfs.status) ~ rx, data = demo.data)
os <- survfit( Surv(os.time, os.status) ~ rx, data = demo.data)
# Combine on the same plot
fit <- list(PFS = pfs, OS = os)
ggsurvplot_combine(fit, demo.data)
```

`ggsurvplot_df` *Plot Survival Curves from Survival Summary Data Frame*

Description

An extension to ggsurvplot() to plot survival curves from any data frame containing the summary of survival curves as returned the surv_summary() function.

Might be useful for a user who wants to use ggsurvplot for visualizing survival curves computed by another method than the standard survfit.formula function. In this case, the user has just to provide the data frame containing the summary of the survival analysis.

Usage

```
ggsurvplot_df(
  fit,
  fun = NULL,
  color = NULL,
```

```
  palette = NULL,
  linetype = 1,
  break.x.by = NULL,
  break.time.by = NULL,
  break.y.by = NULL,
  surv.scale = c("default", "percent"),
  surv.geom = geom_step,
  xscale = 1,
  conf.int = FALSE,
  conf.int.fill = "gray",
  conf.int.style = "ribbon",
  conf.int.alpha = 0.3,
  censor = TRUE,
  censor.shape = "+",
  censor.size = 4.5,
  title = NULL,
  xlab = "Time",
  ylab = "Survival probability",
  xlim = NULL,
  ylim = NULL,
  axes.offset = TRUE,
  legend = c("top", "bottom", "left", "right", "none"),
  legend.title = "Strata",
  legend.labs = NULL,
  ggtheme = theme_survminer(),
  ...
)
```

Arguments

fit	a data frame as returned by surv_summary. Should contains at least the following columns: • time: survival time • surv: survival probability • strata: grouping variables • n.censor: number of censors • upper: upper end of confidence interval • lower: lower end of confidence interval
fun	an arbitrary function defining a transformation of the survival curve. Often used transformations can be specified with a character argument: "event" plots cumulative events (f(y) = 1-y), "cumhaz" plots the cumulative hazard function (f(y) = -log(y)), and "pct" for survival probability in percentage.
color	color to be used for the survival curves. • If the number of strata/group (n.strata) = 1, the expected value is the color name. For example color = "blue". • If n.strata > 1, the expected value is the grouping variable name. By default, survival curves are colored by strata using the argument color = "strata", but

you can also color survival curves by any other grouping variables used to fit the survival curves. In this case, it's possible to specify a custom color palette by using the argument palette.

`palette` the color palette to be used. Allowed values include "hue" for the default hue color scale; "grey" for grey color palettes; brewer palettes e.g. "RdBu", "Blues", ...; or custom color palette e.g. c("blue", "red"); and scientific journal palettes from ggsci R package, e.g.: "npg", "aaas", "lancet", "jco", "ucscgb", "uchicago", "simpsons" and "rickandmorty". See details section for more information. Can be also a numeric vector of length(groups); in this case a basic color palette is created using the function palette.

`linetype` line types. Allowed values includes i) "strata" for changing linetypes by strata (i.e. groups); ii) a numeric vector (e.g., c(1, 2)) or a character vector c("solid", "dashed").

`break.x.by` alias of break.time.by. Numeric value controlling x axis breaks. Default value is NULL.

`break.time.by` numeric value controlling time axis breaks. Default value is NULL.

`break.y.by` same as break.x.by but for y axis.

`surv.scale` scale transformation of survival curves. Allowed values are "default" or "percent".

`surv.geom` survival curve style. Is the survival curve entered a step function (geom_step) or a smooth function (geom_line).

`xscale` numeric or character value specifying x-axis scale.

- If numeric, the value is used to divide the labels on the x axis. For example, a value of 365.25 will give labels in years instead of the original days.
- If character, allowed options include one of c("d_m", "d_y", "m_d", "m_y", "y_d", "y_m"), where d = days, m = months and y = years. For example, xscale = "d_m" will transform labels from days to months; xscale = "m_y", will transform labels from months to years.

`conf.int` logical value. If TRUE, plots confidence interval.

`conf.int.fill` fill color to be used for confidence interval.

`conf.int.style` confidence interval style. Allowed values include c("ribbon", "step").

`conf.int.alpha` numeric value specifying fill color transparency. Value should be in [0, 1], where 0 is full transparency and 1 is no transparency.

`censor` logical value. If TRUE, censors will be drawn.

`censor.shape` character or numeric value specifying the point shape of censors. Default value is "+" (3), a sensible choice is "|" (124).

`censor.size` numveric value specifying the point size of censors. Default is 4.5.

`title` main title and axis labels

`xlab` main title and axis labels

`ylab` main title and axis labels

`xlim` x and y axis limits e.g. xlim = c(0, 1000), ylim = c(0, 1).

`ylim` x and y axis limits e.g. xlim = c(0, 1000), ylim = c(0, 1).

axes.offset	logical value. Default is TRUE. If FALSE, set the plot axes to start at the origin.
legend	character specifying legend position. Allowed values are one of c("top", "bottom", "left", "right", "none"). Default is "top" side position. to remove the legend use legend = "none". Legend position can be also specified using a numeric vector c(x, y); see details section.
legend.title	legend title.
legend.labs	character vector specifying legend labels. Used to replace the names of the strata from the fit. Should be given in the same order as those strata.
ggtheme	function, ggplot2 theme name. Default value is theme_survminer. Allowed values include ggplot2 official themes: see theme.
...	other arguments to be passed i) to ggplot2 geom_*() functions such as linetype, size, ii) or to the function ggpar() for customizing the plots. See details section.

Examples

```
library(survival)

# Fit survival curves
#:::::::::::::::::::::::::::::::::::::::::::::::::::::::::
fit1 <- survfit( Surv(time, status) ~ 1, data = colon)
fit2 <- survfit( Surv(time, status) ~ adhere, data = colon)

# Summary
#:::::::::::::::::::::::::::::::::::::::::::::::::::::::::
head(surv_summary(fit1, colon))

head(surv_summary(fit2, colon))

# Visualize
#:::::::::::::::::::::::::::::::::::::::::::::::::::::::::
ggsurvplot_df(surv_summary(fit1, colon))

ggsurvplot_df(surv_summary(fit2, colon), conf.int = TRUE,
             legend.title = "Adhere", legend.labs = c("0", "1"))

# Kaplan-Meier estimate
#:::::::::::::::::::::::::::::::::::::::::::::::::::::::::
out_km <- survfit(Surv(time, status) ~ 1, data = lung)

# Weibull model
#:::::::::::::::::::::::::::::::::::::::::::::::::::::::::
wb <- survreg(Surv(time, status) ~ 1, data = lung)
s <- seq(.01, .99, by = .01)
t <- predict(wb, type = "quantile", p = s, newdata = lung[1, ])
out_wb <- data.frame(time = t, surv = 1 - s, upper = NA, lower = NA, std.err = NA)

# plot both
#:::::::::::::::::::::::::::::::::::::::::::::::::::::::::
p_km <- ggsurvplot(out_km, conf.int = FALSE)
p_wb <- ggsurvplot(out_wb, conf.int = FALSE, surv.geom = geom_line)
```

```
p_km
p_wb
p_km$plot + geom_line(data = out_wb, aes(x = time, y = surv))
```

---

ggsurvplot_facet *Facet Survival Curves into Multiple Panels*

Description

Draw multi-panel survival curves of a data set grouped by one or two variables.

Usage

```
ggsurvplot_facet(
  fit,
  data,
  facet.by,
  color = NULL,
  palette = NULL,
  legend.labs = NULL,
  pval = FALSE,
  pval.method = FALSE,
  pval.coord = NULL,
  pval.method.coord = NULL,
  nrow = NULL,
  ncol = NULL,
  scales = "fixed",
  short.panel.labs = FALSE,
  panel.labs = NULL,
  panel.labs.background = list(color = NULL, fill = NULL),
  panel.labs.font = list(face = NULL, color = NULL, size = NULL, angle = NULL),
  panel.labs.font.x = panel.labs.font,
  panel.labs.font.y = panel.labs.font,
  ...
)
```

Arguments

fit	an object of class survfit.
data	a dataset used to fit survival curves. If not supplied then data will be extracted from 'fit' object.
facet.by	character vector, of length 1 or 2, specifying grouping variables for faceting the plot. Should be in the data.
color	color to be used for the survival curves.

- If the number of strata/group (n.strata) = 1, the expected value is the color name. For example color = "blue".
- If n.strata > 1, the expected value is the grouping variable name. By default, survival curves are colored by strata using the argument color = "strata", but you can also color survival curves by any other grouping variables used to fit the survival curves. In this case, it's possible to specify a custom color palette by using the argument palette.

`palette` the color palette to be used. Allowed values include "hue" for the default hue color scale; "grey" for grey color palettes; brewer palettes e.g. "RdBu", "Blues", ...; or custom color palette e.g. c("blue", "red"); and scientific journal palettes from ggsci R package, e.g.: "npg", "aaas", "lancet", "jco", "ucscgb", "uchicago", "simpsons" and "rickandmorty". See details section for more information. Can be also a numeric vector of length(groups); in this case a basic color palette is created using the function palette.

`legend.labs` character vector specifying legend labels. Used to replace the names of the strata from the fit. Should be given in the same order as those strata.

`pval` logical value, a numeric or a string. If logical and TRUE, the p-value is added on the plot. If numeric, than the computet p-value is substituted with the one passed with this parameter. If character, then the customized string appears on the plot. See examples - Example 3.

`pval.method` whether to add a text with the test name used for calculating the pvalue, that corresponds to survival curves' comparison - used only when `pval=TRUE`

`pval.coord` numeric vector, of length 2, specifying the x and y coordinates of the p-value. Default values are NULL.

`pval.method.coord` the same as `pval.coord` but for displaying `log.rank.weights` name

`nrow, ncol` Number of rows and columns in the pannel. Used only when the data is faceted by one grouping variable.

`scales` should axis scales of panels be fixed ("fixed", the default), free ("free"), or free in one dimension ("free_x", "free_y").

`short.panel.labs` logical value. Default is FALSE. If TRUE, create short labels for panels by omitting variable names; in other words panels will be labelled only by variable grouping levels.

`panel.labs` a list of one or two character vectors to modify facet label text. For example, panel.labs = list(sex = c("Male", "Female")) specifies the labels for the "sex" variable. For two grouping variables, you can use for example panel.labs = list(sex = c("Male", "Female"), rx = c("Obs", "Lev", "Lev2")).

`panel.labs.background` a list to customize the background of panel labels. Should contain the combination of the following elements:

- `color,linetype,size`: background line color, type and size
- `fill`: background fill color.

For example, panel.labs.background = list(color = "blue", fill = "pink").

panel.labs.font
: a list of aestheics indicating the size (e.g.: 14), the face/style (e.g.: "plain", "bold", "italic", "bold.italic") and the color (e.g.: "red") and the orientation angle (e.g.: 45) of panel labels.

panel.labs.font.x, panel.labs.font.y
: same as panel.labs.font but for x and y direction, respectively.

...
: other arguments to pass to the function ggsurvplot.

Examples

```
library(survival)

# Facet by one grouping variables: rx
#:::::::::::::::::::::::::::::::::::::::::::::::::::::::::
fit <- survfit( Surv(time, status) ~ sex, data = colon )
ggsurvplot_facet(fit, colon, facet.by = "rx",
                 palette = "jco", pval = TRUE)

# Facet by two grouping variables: rx and adhere
#:::::::::::::::::::::::::::::::::::::::::::::::::::::::::
ggsurvplot_facet(fit, colon, facet.by = c("rx", "adhere"),
                 palette = "jco", pval = TRUE)


# Another fit
#:::::::::::::::::::::::::::::::::::::::::::::::::::::::::
fit2 <- survfit( Surv(time, status) ~ sex + rx, data = colon )
ggsurvplot_facet(fit2, colon, facet.by = "adhere",
                 palette = "jco", pval = TRUE)
```

---

ggsurvplot_group_by *Survival Curves of Grouped Data sets*

Description

Survival curves of grouped data sets by one or two variables.

Survival analysis are often done on subsets defined by variables in the dataset. For example, assume that we have a cohort of patients with a large number of clinicopathological and molecular covariates, including survival data, TP53 mutation status and the patients' sex (Male or Female).

One might be also interested in comparing the survival curves of Male and Female after grouping (or splitting) the data by TP53 mutation status.

ggsurvplot_group_by() provides a convenient solution to create a multiple ggsurvplot of a data set grouped by one or two variables.

Usage

```
ggsurvplot_group_by(fit, data, group.by, ...)
```

Arguments

fit	a survfit object.
data	a data frame used to fit survival curves.
group.by	a character vector containing the name of grouping variables. Should be of length <= 2.
...	... other arguments passed to the core function ggsurvplot.

Details

ggsurvplot_group_by() works as follow:

1. Create a grouped data sets using the function surv_group_by(), –> list of data sets
2. Map surv_fit() to each nested data –> Returns a list of survfit objects
3. Map ggsurvplot() to each survfit object –> list of survfit ggsurvplots

One can (optionally) arrange the list of ggsurvplots using arrange_ggsurvplots()

Value

Retuns a list of ggsurvplots.

Examples

```
# Fit survival curves
#:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
library(survival)
fit <- survfit( Surv(time, status) ~ sex, data = colon )

# Visualize: grouped by treatment rx
#:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
ggsurv.list <- ggsurvplot_group_by(fit, colon, group.by = "rx", risk.table = TRUE,
                                   pval = TRUE, conf.int = TRUE, palette = "jco")
names(ggsurv.list)


# Visualize: grouped by treatment rx and adhere
#:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
ggsurv.list <- ggsurvplot_group_by(fit, colon, group.by = c("rx", "adhere"),
                                   risk.table = TRUE,
                                   pval = TRUE, conf.int = TRUE, palette = "jco")

names(ggsurv.list)
```

ggsurvplot_list *Plot a List of Survfit Objects*

Description

Take a list of survfit objects and produce a list of ggsurvplots.

Usage

```
ggsurvplot_list(
  fit,
  data,
  title = NULL,
  legend.labs = NULL,
  legend.title = "Strata",
  ...
)
```

Arguments

fit	a list of survfit objects.
data	data used to fit survival curves. Can be also a list of same length than fit.
title	title of the plot. Can be a character vector or a list of titles of same length than fit. If title is not specified and fit is a named list, then the names of fit list are used as title.
legend.labs	character vector specifying legend labels. Used to replace the names of the strata from the fit. Should be given in the same order as those strata. Can be a list when fit is a list.
legend.title	legend title for each plot. Can be a character vector or a list of titles of same length than fit.
...	other arguments passed to the core function ggsurvplot

Value

Returns a list of ggsurvplots.

See Also

ggsurvplot

Examples

```
library(survival)

# Create a list of formulas
```

```
#::::::::::::::::::::::::::::::::::::::::::::::::::::
data(colon)
f1 <- survfit(Surv(time, status) ~ adhere, data = colon)
f2 <- survfit(Surv(time, status) ~ rx, data = colon)
fits <- list(sex = f1, rx = f2)

# Visualize
#::::::::::::::::::::::::::::::::::::::::::::::::::::
legend.title <- list("sex", "rx")
ggsurvplot_list(fits, colon, legend.title = legend.title)
```

`myeloma` *Multiple Myeloma Data*

Description

Multiple Myeloma data extracted from publicly available gene expression data (GEO Id: GSE4581).

Usage

```
data("myeloma")
```

Format

A data frame with 256 rows and 12 columns.

`molecular_group` Patients' molecular subgroups

`chr1q21_status` Amplification status of the chromosome 1q21

`treatment` treatment

`event` survival status 0 = alive, 1 = dead

`time` Survival time in months

`CCND1` Gene expression

`CRIM1` Gene expression

`DEPDC1` Gene expression

`IRF4` Gene expression

`TP53` Gene expression

`WHSC1` Gene expression

The remaining columns (CCND1, CRIM1, DEPDC1, IRF4, TP53, WHSC1) correspond to the gene expression level of specified genes.

Examples

```
data(myeloma)
head(myeloma)
```

pairwise_survdiff *Multiple Comparisons of Survival Curves*

Description

Calculate pairwise comparisons between group levels with corrections for multiple testing.

Usage

```
pairwise_survdiff(formula, data, p.adjust.method = "BH", na.action, rho = 0)
```

Arguments

formula	a formula expression as for other survival models, of the form Surv(time, status) ~ predictors.
data	a data frame in which to interpret the variables occurring in the formula.
p.adjust.method	method for adjusting p values (see p.adjust). Allowed values include "holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr", "none". If you don't want to adjust the p value (not recommended), use p.adjust.method = "none".
na.action	a missing-data filter function. Default is options()$na.action.
rho	a scalar parameter that controls the type of test. Allowed values include 0 (for Log-Rank test) and 1 (for peto & peto test).

Value

Returns an object of class "pairwise.htest", which is a list containing the p values.

Author(s)

Alboukadel Kassambara, <alboukadel.kassambara@gmail.com>

See Also

survival::survdiff

Examples

```
library(survival)
library(survminer)
data(myeloma)

# Pairwise survdiff
res <- pairwise_survdiff(Surv(time, event) ~ molecular_group,
     data = myeloma)
res
```

```
# Symbolic number coding
symnum(res$p.value, cutpoints = c(0, 0.0001, 0.001, 0.01, 0.05, 0.1, 1),
   symbols = c("****", "***", "**", "*", "+", " "),
   abbr.colnames = FALSE, na = "")
```

---

surv_cutpoint *Determine the Optimal Cutpoint for Continuous Variables*

Description

Determine the optimal cutpoint for one or multiple continuous variables at once, using the maximally selected rank statistics from the 'maxstat' R package. This is an outcome-oriented methods providing a value of a cutpoint that correspond to the most significant relation with outcome (here, survival).

- `surv_cutpoint()`: Determine the optimal cutpoint for each variable using 'maxstat'.
- `surv_categorize()`: Divide each variable values based on the cutpoint returned by `surv_cutpoint()`.

Usage

```
surv_cutpoint(
  data,
  time = "time",
  event = "event",
  variables,
  minprop = 0.1,
  progressbar = TRUE
)

surv_categorize(x, variables = NULL, labels = c("low", "high"))

## S3 method for class 'surv_cutpoint'
summary(object, ...)

## S3 method for class 'surv_cutpoint'
print(x, ...)

## S3 method for class 'surv_cutpoint'
plot(x, variables = NULL, ggtheme = theme_classic(), bins = 30, ...)

## S3 method for class 'plot_surv_cutpoint'
print(x, ..., newpage = TRUE)
```

Arguments

data	a data frame containing survival information (time, event) and continuous variables (e.g.: gene expression data).
time, event	column names containing time and event data, respectively. Event values sould be 0 or 1.
variables	a character vector containing the names of variables of interest, for wich we want to estimate the optimal cutpoint.
minprop	the minimal proportion of observations per group.
progressbar	logical value. If TRUE, show progress bar. Progressbar is shown only, when the number of variables > 5.
x, object	an object of class surv_cutpoint
labels	labels for the levels of the resulting category.
...	other arguments. For plots, see ?ggpubr::ggpar
ggtheme	function, ggplot2 theme name. Default value is theme_classic. Allowed values include ggplot2 official themes. see ?ggplot2::ggtheme.
bins	Number of bins for histogram. Defaults to 30.
newpage	open a new page. See grid.arrange.

Value

- **surv_cutpoint()**: returns an object of class 'surv_cutpoint', which is a list with the following components:
  - maxstat results for each variable (see ?maxstat::maxstat)
  - cutpoint: a data frame containing the optimal cutpoint of each variable. Rows are variable names and columns are c("cutpoint", "statistic").
  - data: a data frame containing the survival data and the original data for the specified variables.
  - minprop: the minimal proportion of observations per group.
  - not_numeric: contains data for non-numeric variables, in the context where the user provided categorical variable names in the argument variables.

  Methods defined for surv_cutpoint object are summary, print and plot.
- **surv_categorize()**: returns an object of class 'surv_categorize', which is a data frame containing the survival data and the categorized variables.

Author(s)

Alboukadel Kassambara, <alboukadel.kassambara@gmail.com>

Examples

```
# 0. Load some data
data(myeloma)
head(myeloma)
```

```
# 1. Determine the optimal cutpoint of variables
res.cut <- surv_cutpoint(myeloma, time = "time", event = "event",
   variables = c("DEPDC1", "WHSC1", "CRIM1"))

summary(res.cut)

# 2. Plot cutpoint for DEPDC1
# palette = "npg" (nature publishing group), see ?ggpubr::ggpar
plot(res.cut, "DEPDC1", palette = "npg")

# 3. Categorize variables
res.cat <- surv_categorize(res.cut)
head(res.cat)

# 4. Fit survival curves and visualize
library("survival")
fit <- survfit(Surv(time, event) ~DEPDC1, data = res.cat)
ggsurvplot(fit, data = res.cat, risk.table = TRUE, conf.int = TRUE)
```

---

surv_fit *Create Survival Curves*

Description

Wrapper arround the standard survfit() function to create survival curves. Compared to the standard survfit() function, it supports also:

- a list of data sets and/or a list of formulas,
- a grouped data sets as generated by the function surv_group_by,
- group.by option

There are many cases, where this function might be useful:

- **Case 1: One formula and One data set**. Example: You want to fit the survival curves of one biomarker/gene in a given data set. This is the same as the standard survfit() function. Returns one survfit object.
- **Case 2: List of formulas and One data set**. Example: You want to fit the survival curves of a list of biormarkers/genes in the same data set. Returns a named list of survfit objects in the same order as formulas.
- **Case 3: One formula and List of data sets**. Example: You want to fit survival curves of one biomarker/gene in multiple cohort of patients (colon, lung, breast). Returns a named list of survfit objects in the same order as the data sets.
- **Case 4: List of formulas and List of data sets**. Example: You want to fit survival curves of multiple biomarkers/genes in multiple cohort of patients (colon, lung, breast). Each formula will be applied to each of the data set in the data list. Returns a named list of survfit objects.

- **Case 5: One formula and grouped data sets by one or two variables**. Example: One might like to plot the survival curves of patients treated by drug A vs patients treated by drug B in a dataset grouped by TP53 and/or RAS mutations. In this case use the argument `group.by`. Returns a named list of survfit objects.
- **Case 6**. In a rare case you might have a list of formulas and a list of data sets, and you might want to **apply each formula to the mathcing data set with the same index/position in the list**. For example formula1 is applied to data 1, formula2 is applied to data 2, and so on ... In this case formula and data lists should have the same length and you should specify the argument match.fd = TRUE (stands for match formula and data). Returns a named list of survfit objects.

The output of the `surv_fit()` function can be directly handled by the following functions:

- ggsurvplot()
- surv_pvalue()
- surv_median()

These functions return one element or a list of elements depending on the format of the input.

Usage

```
surv_fit(formula, data, group.by = NULL, match.fd = FALSE, ...)
```

Arguments

`formula`	survival formula. See survfit.formula. Can be a list of formula. Named lists are recommended.
`data`	a data frame in which to interpret the variables named in the formula. Can be a list of data sets. Named lists are recommended. Can be also a grouped dataset as generated by the function surv_group_by().
`group.by`	a grouping variables to group the data set by. A character vector containing the name of grouping variables. Should be of length <= 2.
`match.fd`	logical value. Default is FALSE. Stands for "match formula and data". Useful only when you have a list of formulas and a list of data sets, and you want to apply each formula to the matching data set with the same index/position in the list. For example formula1 is applied to data 1, formula2 is applied to data 2, and so on .... In this case use match.fd = TRUE.
`...`	Other arguments passed to the survfit.formula function.

Value

- Returns an object of class survfit if one formula and one data set provided.
- Returns a named list of survfit objects when input is a list of formulas and/or data sets. The same holds true when grouped data sets are provided or when the argument `group.by` is specified.
  - If the names of formula and data lists are available, the names of the resulting survfit objects list are obtained by collapsing the names of formula and data lists.

- If the formula names are not available, the variables in the formulas are extracted and used to build the name of survfit object.
- In the case of grouped data sets, the names of survfit object list are obtained by collapsing the levels of grouping variables and the names of variables in the survival curve formulas.

Examples

```
library("survival")
library("magrittr")

# Case 1: One formula and One data set
#:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
fit <- surv_fit(Surv(time, status) ~ sex,
               data = colon)
surv_pvalue(fit)


# Case 2: List of formulas and One data set.
#   - Different formulas are applied to the same data set
#   - Returns a (named) list of survfit objects
#:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
# Create a named list of formulas
formulas <- list(
 sex = Surv(time, status) ~ sex,
 rx = Surv(time, status) ~ rx
)

# Fit survival curves for each formula
fit <- surv_fit(formulas, data = colon)
surv_pvalue(fit)

# Case 3: One formula and List of data sets
#:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
fit <- surv_fit(Surv(time, status) ~ sex,
               data = list(colon, lung))
surv_pvalue(fit)


# Case 4: List of formulas and List of data sets
#  - Each formula is applied to each of the data in the data list
#  - argument: match.fd = FALSE
#:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

# Create two data sets
set.seed(123)
colon1 <- dplyr::sample_frac(colon, 1/2)
set.seed(1234)
colon2 <- dplyr::sample_frac(colon, 1/2)

# Create a named list of formulas
formula.list <- list(
```

```
 sex = Surv(time, status) ~ sex,
 adhere = Surv(time, status) ~ adhere,
 rx = Surv(time, status) ~ rx
)

# Fit survival curves
fit <- surv_fit(formula.list, data = list(colon1, colon2),
               match.fd = FALSE)
surv_pvalue(fit)


# Grouped survfit
#:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
# - Group by the treatment "rx" and fit survival curves on each subset
# - Returns a list of survfit objects
fit <- surv_fit(Surv(time, status) ~ sex,
               data = colon, group.by = "rx")

# Alternatively, do this
fit <- colon %>%
 surv_group_by("rx") %>%
 surv_fit(Surv(time, status) ~ sex, data = .)

surv_pvalue(fit)
```

`surv_group_by` *Create a Grouped Dataset for Survival Analysis*

Description

Split a data frame into multiple new data frames based on one or two grouping variables. The `surv_group_by()` function takes an existing data frame and converts it into a grouped data frame where survival analysis are performed "by group".

Usage

```
surv_group_by(data, grouping.vars)
```

Arguments

`data`	a data frame
`grouping.vars`	a character vector containing the name of grouping variables. Should be of length <= 2

Value

Returns an object of class surv_group_by which is a tibble data frame with the following components:

- one column for each grouping variables. Contains the levels.
- a coumn named "data", which is a named list of data subsets created by the grouping variables. The list names are created by concatening the levels of grouping variables.

Examples

```
library("survival")
library("magrittr")

# Grouping by one variables: treatment "rx"
#:::::::::::::::::::::::::::::::::::::::::
grouped.d <- colon %>%
  surv_group_by("rx")

grouped.d # print

grouped.d$data # Access to the data

# Grouping by two variables
#:::::::::::::::::::::::::::::::::::::::::
grouped.d <- colon %>%
   surv_group_by(grouping.vars = c("rx", "adhere"))
   grouped.d
```

---

surv_median *Median of Survival Curves*

Description

Returns the median survival with upper and lower confidence limits for the median at 95% confidence levels.

Usage

```
surv_median(fit, combine = FALSE)
```

Arguments

fit	A survfit object. Can be also a list of survfit objects.
combine	logical value. Used only when fit is a list of survfit objects. If TRUE, combine the results for multiple fits.

Value

Returns for each fit, a data frame with the following column:

- strata: strata/group names
- median: median survival of each group
- lower: 95% lower confidence limit
- upper: 95% upper confidence limit

Returns a list of data frames when the input is a list of survfit objects. If combine = TRUE, results are combined into one single data frame.

Examples

```
library(survival)

# Different survfits
#:::::::::::::::::::::::::::::::::::::::::::::::::::::::
fit.null <- surv_fit(Surv(time, status) ~ 1, data = colon)

fit1 <- surv_fit(Surv(time, status) ~ sex, data = colon)

fit2 <- surv_fit(Surv(time, status) ~ adhere, data = colon)

fit.list <- list(sex = fit1, adhere = fit2)

# Extract the median survival
#:::::::::::::::::::::::::::::::::::::::::::::::::::::::
surv_median(fit.null)

surv_median(fit2)

surv_median(fit.list)

surv_median(fit.list, combine = TRUE)

# Grouped survfit
#:::::::::::::::::::::::::::::::::::::::::::::::::::::::
fit.list2 <- surv_fit(Surv(time, status) ~ sex, data = colon,
                      group.by = "rx")
surv_median(fit.list2)
```

---

surv_pvalue *Compute P-value Comparing Survival Curves*

Description

Compute p-value from survfit objects or parse it when provided by the user. Survival curves are compared using the log-rank test (default). Other methods can be specified using the argument `method`.

Usage

```
surv_pvalue(
  fit,
  data = NULL,
  method = "survdiff",
  test.for.trend = FALSE,
  combine = FALSE,
  ...
)
```

Arguments

fit	A survfit object. Can be also a list of survfit objects.
data	data frame used to fit survival curves. Can be also a list of data.
method	method to compute survival curves. Default is "survdiff" (or "log-rank"). Allowed values are one of: • "survdiff", log-rank; • "1": log-rank, LR; –> Regular log-rank test, sensitive to detect late differences. • "n": Gehan-Breslow (generalized Wilcoxon), GB; –> detect early differences. • "sqrtN": Tarone-Ware, TW; –> detect early differences. • "S1": Peto-Peto's modified survival estimate, PP; –> more robust than Tharone-Whare or Gehan-Breslow, detect early differences • "S2": modified Peto-Peto (by Andersen), mPP • "FH_p=1_q=1": Fleming-Harrington(p=1, q=1), FH To specify method, one can use either the weights (e.g.: "1", "n", "sqrtN", ...), or the full name ("log-rank", "gehan-breslow", "Peto-Peto", ...), or the acronyme LR, GB, ... Case insensitive partial match is allowed. To learn more about the mathematical background behind the different log-rank weights, read the following blog post on R-Addict: Comparing (Fancy) Survival Curves with Weighted Log-rank Tests
test.for.trend	logical value. Default is FALSE. If TRUE, returns the test for trend p-values. Tests for trend are designed to detect ordered differences in survival curves. That is, for at least one group. The test for trend can be only performed when the number of groups is > 2.
combine	logical value. Used only when fit is a list of survfit objects. If TRUE, combine the results for multiple fits.
...	other arguments including pval, pval.coord, pval.method.coord. These are only used internally to specify custom pvalue, pvalue and pvalue method coordinates on the survival plot. Normally, users don't need these arguments.

Value

Return a data frame with the columns (pval, method, pval.txt and variable). If additional arguments (pval, pval.coord, pval.method.coord, get_coord) are specified, then extra columns (pval.x, pval.y, method.x and method.y) are returned.

- pval: pvalue
- method: method used to compute pvalues
- pval.txt: formatted text ready to use for annotating plots
- pval.x, pval.y: x & y coordinates of the pvalue for annotating the plot
- method.x, method.y: x & y coordinates of pvalue method

Examples

```
library(survival)

# Different survfits
#:::::::::::::::::::::::::::::::::::::::::::::::::::::::
fit.null <- surv_fit(Surv(time, status) ~ 1, data = colon)

fit1 <- surv_fit(Surv(time, status) ~ sex, data = colon)

fit2 <- surv_fit(Surv(time, status) ~ adhere, data = colon)

fit.list <- list(sex = fit1, adhere = fit2)

# Extract the median survival
#:::::::::::::::::::::::::::::::::::::::::::::::::::::::
surv_pvalue(fit.null)

surv_pvalue(fit2, colon)

surv_pvalue(fit.list)

surv_pvalue(fit.list, combine = TRUE)

# Grouped survfit
#:::::::::::::::::::::::::::::::::::::::::::::::::::::::
fit.list2 <- surv_fit(Surv(time, status) ~ sex, data = colon,
                      group.by = "rx")

surv_pvalue(fit.list2)

# Get coordinate for annotion of the survival plots
#:::::::::::::::::::::::::::::::::::::::::::::::::::::::
surv_pvalue(fit.list2, combine = TRUE, get_coord = TRUE)
```

surv_summary *Nice Summary of a Survival Curve*

Description

Compared to the default summary() function, surv_summary() creates a data frame containing a nice summary from survfit results.

Usage

```
surv_summary(x, data = NULL)
```

Arguments

x	an object of class survfit.
data	a dataset used to fit survival curves. If not supplied then data will be extracted from 'fit' object.

Value

An object of class **'surv_summary'**, which is a data frame with the following columns:

- time: the time points at which the curve has a step.
- n.risk: the number of subjects at risk at t.
- n.event: the number of events that occur at time t.
- n.censor: number of censored events.
- surv: estimate of survival.
- std.err: standard error of survival.
- upper: upper end of confidence interval.
- lower: lower end of confidence interval.
- strata: stratification of survival curves.

In a situation, where survival curves have been fitted with one or more variables, surv_summary object contains **extra columns** representing the variables. This makes it possible to facet the output of ggsurvplot by strata or by some combinations of factors.

surv_summary object has also an attribut named **'table'** containing information about the survival curves, including medians of survival with confidence intervals, as well as, the total number of subjects and the number of event in each curve.

Author(s)

Alboukadel Kassambara, <alboukadel.kassambara@gmail.com>

Examples

```
# Fit survival curves
require("survival")
fit <- survfit(Surv(time, status) ~ rx + adhere, data = colon)

# Summarize
res.sum <- surv_summary(fit, data = colon)
head(res.sum)

# Information about the survival curves
attr(res.sum, "table")
```

`theme_survminer` *Theme for Survminer Plots*

Description

Default theme for plots generated with survminer.

Usage

```
theme_survminer(
  base_size = 12,
  base_family = "",
  font.main = c(16, "plain", "black"),
  font.submain = c(15, "plain", "black"),
  font.x = c(14, "plain", "black"),
  font.y = c(14, "plain", "black"),
  font.caption = c(15, "plain", "black"),
  font.tickslab = c(12, "plain", "black"),
  legend = c("top", "bottom", "left", "right", "none"),
  font.legend = c(10, "plain", "black"),
  ...
)

theme_cleantable(base_size = 12, base_family = "", ...)
```

Arguments

`base_size`	base font size
`base_family`	base font family
`font.main`, `font.submain`, `font.caption`, `font.x`, `font.y`, `font.tickslab`, `font.legend`	a vector of length 3 indicating respectively the size (e.g.: 14), the style (e.g.: "plain", "bold", "italic", "bold.italic") and the color (e.g.: "red") of main title,

	subtitle, caption, xlab and ylab, axis tick labels and legend, respectively. For example *font.x = c(14, "bold", "red")*. Use font.x = 14, to change only font size; or use font.x = "bold", to change only font face.
legend	character specifying legend position. Allowed values are one of c("top", "bottom", "left", "right", "none"). Default is "top" side position. to remove the legend use legend = "none". Legend position can be also specified using a numeric vector c(x, y); see details section.
...	additional arguments passed to the function theme_survminer().

Functions

- `theme_survminer`: Default theme for survminer plots. A theme similar to theme_classic() with large font size.
- `theme_cleantable`: theme for drawing a clean risk table and cumulative number of events table. A theme similar to theme_survminer() without i) axis lines and, ii) x axis ticks and title.

Author(s)

Alboukadel Kassambara, <alboukadel.kassambara@gmail.com>

Examples

```
# Fit survival curves
#++++++++++++++++++++++++++++++++++
require("survival")
fit<- survfit(Surv(time, status) ~ sex, data = lung)

# Basic survival curves
#++++++++++++++++++++++++++++++++++
ggsurv <- ggsurvplot(fit, data = lung, risk.table = TRUE,
   main = "Survival curves",
   submain = "Based on Kaplan-Meier estimates",
   caption = "created with survminer"
   )

# Change font size, style and color
#++++++++++++++++++++++++++++++++++
# Change font size, style and color at the same time
# Use font.x = 14, to change only font size; or use
# font.x = "bold", to change only font face.
ggsurv %+% theme_survminer(
     font.main = c(16, "bold", "darkblue"),
     font.submain = c(15, "bold.italic", "purple"),
     font.caption = c(14, "plain", "orange"),
     font.x = c(14, "bold.italic", "red"),
     font.y = c(14, "bold.italic", "darkred"),
     font.tickslab = c(12, "plain", "darkgreen")
   )

# Clean risk table
```

```
# ++++++++++++++++++++++++++++
ggsurv$table <- ggsurv$table + theme_cleantable()
ggsurv
```

Index