

Package: survminer (via r-universe)

September 29, 2026

Type Package

Title Drawing Survival Curves using 'ggplot2'

Version 0.5.2.999

Date 2026-02-25

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(>= 3.4.0), ggpubr(>= 0.1.6), R (>= 2.10)

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

Suggests knitr, flexsurv, cmprsk, tidycmprsk, markdown, testthat, rmarkdown, survRM2, prodlim

VignetteBuilder knitr

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

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

RoxygenNote 7.3.3

Collate 'BMT.R' 'BRCAOV.survInfo.R' 'add_ggsurvplot.R' 'customize_labels.R' 'adtte.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' 'ggcoxnph.R' 'ggcoxzph.R' 'ggflexsurvplot.R' 'ggforest.R' 'ggforest_subgroup.R' 'ggforest_models.R' 'gglandmark.R' 'ggmilestone.R' 'ggmst.R'

```
'ggsurvevents.R' 'ggsurvparametric.R' 'ggsurvplot_combine.R'
'ggsurvplot_facet.R' 'ggsurvtheme.R' 'ggsurvplot_arguments.R'
'myeloma.R' 'pairwise_survdif.R' 'presets.R' 'surv_adtte.R'
'surv_cutpoint.R' 'surv_group_by.R' 'surv_fit.R'
'surv_median.R' 'surv_median_followup.R' 'weighted_logrank.R'
```

Config/pak/sysreqs cmake make libicu-dev libjpeg-dev libpng-dev libxml2-dev libssl-dev

Repository <https://kassambara.r-universe.dev>

Date/Publication 2026-07-29 09:39:23 UTC

RemoteUrl <https://github.com/kassambara/survminer>

RemoteRef HEAD

RemoteSha 86c3aa209b3a7c4689ee731051d1af152930267a

Contents

add_ggsurvplot	3
adtte	4
arrange_ggsurvplots	5
BMT	7
BRCAOV.survInfo	8
build_ggsurvplot	9
customize_labels	10
ggadjustedcurves	11
ggcompetingrisks	16
ggcoxdiagnostics	19
ggcoxfunctional	21
ggcoxnph	23
ggcoxzph	26
ggflexsurvplot	28
ggforest	29
ggforest_models	32
ggforest_subgroup	33
gglandmark	36
ggmilestone	38
ggrisktable	40
ggrmst_difference	44
ggsurvevents	46
ggsurvparametric	48
ggsurvplot	49
ggsurvplot_add_all	58
ggsurvplot_arguments	60
ggsurvplot_combine	64
ggsurvplot_df	66
ggsurvplot_facet	70
ggsurvplot_group_by	74
ggsurvplot_list	76
myeloma	77

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.tickslabel = c(12, "plain", "darkgreen")
)
```

adtte

Example CDISC ADTTE Time-to-Event Dataset

Description

A small, simulated CDISC ADaM time-to-event analysis dataset (ADTTE) for demonstrating `surv_adtte()`. It follows the ADaM structure and the CNSR censoring convention (**0 = event**, ≥ 1 = **censored**), with two analysis parameters and a clear treatment effect. The data are entirely synthetic (no real patients).

Usage

```
data("adtte")
```

Format

A data frame with 520 rows (260 subjects x 2 parameters) and 9 columns.

STUDYID study identifier

USUBJID unique subject identifier

TRT01P planned treatment: "Placebo" or "Active Treatment"

PARAMCD parameter code: "OS" (overall survival) or "PFS" (progression-free survival)

PARAM parameter description

AVAL analysis value: the time to event or censoring, in months

AVALU analysis value unit ("MONTHS")
 CNSR censoring flag: 0 = event, 1 = administratively censored (alive/ongoing at the data cutoff), 2 = lost to follow-up
 EVNTDESC event or censoring description

Details

Simulated with a strong treatment benefit (Active Treatment vs Placebo: median overall survival roughly 34 vs 12 months), so the Kaplan-Meier curves separate clearly (log-rank $p < 0.0001$) for both parameters. Because CNSR uses the CDISC coding (0 = event), it must be converted before use in `Surv()` – which is what `surv_adtte()` does.

See Also

[surv_adtte](#)

Examples

```
data("adtte")
head(adtte)

os <- surv_adtte(adtte, paramcd = "OS")
fit <- surv_fit(survival::Surv(AVAL, event) ~ TRT01P, data = os)
ggsurvplot(fit, data = os, pval = TRUE, risk.table = TRUE)
```

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,
  layout_matrix = NULL,
  ...
)
```

Arguments

<code>x</code>	a list of ggsurvplots.
<code>print</code>	logical value. If TRUE, the arranged plots are displayed.
<code>title</code>	character vector specifying page title. Default is NA.
<code>ncol, nrow</code>	the number of columns and rows, respectively.
<code>surv.plot.height</code>	the height of the survival plot on the grid. Default is 0.75. Ignored when <code>risk.table = FALSE</code> . $1 - \text{risk.table.height} - \text{ncensor.plot.height}$ when <code>risk.table = TRUE</code> and <code>ncensor.plot = TRUE</code>
<code>risk.table.height</code>	the height of the risk table on the grid. Increase the value when you have many strata. Default is 0.25. Ignored when <code>risk.table = FALSE</code> .
<code>ncensor.plot.height</code>	The height of the censor plot. Used when <code>ncensor.plot = TRUE</code> .
<code>layout_matrix</code>	optional layout matrix passed to marrangeGrob to control the position/order of the plots. For example, <code>matrix(seq_along(x), nrow = nrow, byrow = TRUE)</code> orders the plots by row. Default is NULL (plots are filled column-wise).
<code>...</code>	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 preparation
#+++++
# 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

Breast and Ovarian cancers survival information from the RTCGA.clinical R/Bioconductor package. <http://rtcg.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)
```

build_ggsurvplot	<i>Build the Assembled Grob of a ggsurvplot Object</i>
------------------	--

Description

Assemble a [ggsurvplot](#) object (the survival curve plus any risk table, cumulative events/censor tables and censor plot) into a single gtable grob – the same object that `print.ggsurvplot()` draws. This is useful when you want the combined plot as a grob to draw with [grid.draw](#), arrange with other grobs, or save with [ggsave](#) without going through the print method.

Usage

```
build_ggsurvplot(
  x,
  surv.plot.height = NULL,
  risk.table.height = NULL,
  ncensor.plot.height = NULL,
  cumevents.height = NULL,
  ...
)
```

Arguments

<code>x</code>	an object of class <code>ggsurvplot</code> .
<code>surv.plot.height</code> , <code>cumevents.height</code>	<code>risk.table.height</code> , <code>ncensor.plot.height</code>
	optional numeric values (in $[0, 1]$) overriding the relative heights of the survival plot and the tables on the grid. Default <code>NULL</code> keeps the heights stored in the object.
<code>...</code>	not used.

Value

A gtable/grob object (as returned by [ggplotGrob](#) / [arrangeGrob](#)).

Examples

```
library(survival)
fit <- survfit(Surv(time, status) ~ sex, data = lung)
p <- ggsurvplot(fit, data = lung, risk.table = TRUE)

# Assemble to a grob and draw it
g <- build_ggsurvplot(p)
grid::grid.newpage()
grid::grid.draw(g)

# or save directly
# ggplot2::ggsave("survival.png", g, width = 7, height = 6)
```

customize_labels

Customize the Font of ggsurvplot Labels

Description

Set the font (size, style and color) of the plot main title, subtitle, caption, axis titles and axis tick labels of a ggsurvplot object or of any ggplot. Works on the whole `ggsurvplot` object at once (survival curve, risk table and censor plot) or on an individual component (e.g. `ggsurv$table`).

Compared to `ggpar()`, `customize_labels()` sets the labels using `element_markdown()`, so it preserves (and can style) the markdown-formatted risk-table strata labels produced by `ggsurvplot()`.

Usage

```
customize_labels(
  p,
  font.title = NULL,
  font.subtitle = NULL,
  font.caption = NULL,
  font.x = NULL,
  font.y = NULL,
  font.xtickslab = NULL,
  font.ytickslab = NULL
)
```

Arguments

`p` a ggsurvplot object, a ggplot or a list of ggplots.

`font.title`, `font.subtitle`, `font.caption` 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 the plot main title, subtitle and caption, respectively. For example `font.title = c(14, "bold", "red")`. Use `NULL` (default) to leave the corresponding label unchanged.

`font.x`, `font.y` a vector of length 3 indicating respectively the size, the style and the color of the x and y axis titles.

font.xtickslab, font.ytickslab
 a vector of length 3 indicating respectively the size, the style and the color of the x and y axis tick labels.

Value

Returns an object of the same type as the input p (a single ggplot when a ggplot is supplied, otherwise the list/ ggsurvplot with its ggplot components updated).

Examples

```
## Not run:
library(survival)
fit <- survfit(Surv(time, status) ~ sex, data = lung)
ggsurv <- ggsurvplot(fit, data = lung, risk.table = TRUE)

# Apply the same font style to all the components of ggsurv
# (survival curve, risk table and censor part)
ggsurv <- customize_labels(
  ggsurv,
  font.title      = c(16, "bold", "darkblue"),
  font.subtitle   = 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.xtickslab  = c(12, "plain", "darkgreen")
)
ggsurv

# Customize only the risk table
ggsurv$table <- customize_labels(
  ggsurv$table,
  font.title = c(13, "bold.italic", "green")
)
print(ggsurv)

## End(Not run)
```

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/surv-adjustedcurves/>

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(),
  risk.table = FALSE,
  risk.table.height = 0.25,
  risk.table.title = NULL,
  break.time.by = NULL,
  xlim = NULL,
  tables.y.text = TRUE,
  tables.theme = theme_survminer(),
  fontsize = 4.5,
  show.hr = FALSE,
  hr.coord = NULL,
  hr.size = 4,
  ...
)

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

Arguments

<code>fit</code>	an object of class <code>coxph.object</code> - created with <code>coxph</code> function.
<code>variable</code>	a character, name of the grouping variable to be plotted. If not supplied then it will be extracted from the model formula from the <code>strata()</code> component. If there is no <code>strata()</code> component then only a single curve will be plotted - average for the whole population.

<code>data</code>	a dataset for predictions. If not supplied then data will be extracted from the <code>fit</code> object.
<code>reference</code>	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 <code>fit</code> object.
<code>method</code>	a character, describes how the expected survival curves shall be calculated. Possible options: <code>'single'</code> (average for population), <code>'average'</code> (averages for subpopulations), <code>'marginal'</code> , <code>'conditional'</code> (averages for subpopulations after rebalancing). See the Details section for further information.
<code>fun</code>	an arbitrary function defining a transformation of the survival curve. Often used transformations can be specified with a character argument: <code>"event"</code> plots cumulative events ($f(y) = 1 - y$), <code>"cumhaz"</code> plots the cumulative hazard function ($f(y) = -\log(y)$), and <code>"pct"</code> for survival probability in percentage. For <code>surv_adjustedcurves()</code> the transformation is applied to the returned survival column; the default <code>NULL</code> leaves the survival probabilities unchanged.
<code>palette</code>	the color palette to be used. Allowed values include <code>"hue"</code> for the default hue color scale; <code>"grey"</code> for grey color palettes; brewer palettes e.g. <code>"RdBu"</code> , <code>"Blues"</code> , ...; or custom color palette e.g. <code>c("blue", "red")</code> ; and scientific journal palettes from ggsci R package, e.g.: <code>"npg"</code> , <code>"aaas"</code> , <code>"lancet"</code> , <code>"jco"</code> , <code>"ucscgb"</code> , <code>"uchicago"</code> , <code>"simpsons"</code> and <code>"rickandmorty"</code> . See details section for more information. Can be also a numeric vector of <code>length(groups)</code> ; in this case a basic color palette is created using the function palette .
<code>ylab</code>	a label for oy axis.
<code>size</code>	the curve size.
<code>ggtheme</code>	function, ggplot2 theme name. Allowed values include ggplot2 official themes: see theme .
<code>risk.table</code>	logical value (or a valid ggsurvplot() risk-table string). Default is <code>FALSE</code> . If not <code>FALSE</code> , a number-at-risk table is drawn below the adjusted curves and the function returns a ggsurvplot object (printed with <code>print()</code>), instead of the plain ggplot returned by default. The table is the Kaplan-Meier number at risk <i>by the grouping</i> variable: the adjusted curves are model-based expectations for covariate profiles and have no literal number at risk of their own, so this table is meaningful only when variable is a real subgroup. It is computed on the rows the Cox model used (its complete cases), so its baseline count matches <code>fit\$n</code> . For <code>method = "marginal"/"conditional"</code> the curves are balanced/reweighted while this count is the raw Kaplan-Meier number at risk. Note that when a table is drawn the return value is a ggsurvplot object, so <code>post-hoc + theme()/+ scale_*</code> no longer apply – use the arguments below (<code>xlim</code> , <code>break.time.by</code> , <code>ylab</code> , <code>palette</code> , <code>ggtheme</code>) instead.
<code>risk.table.height</code>	the height of the risk table relative to the plot when <code>risk.table</code> is drawn. Default is 0.25.
<code>risk.table.title</code>	the risk-table title. Default <code>NULL</code> uses <code>"Number at risk (Kaplan-Meier, by <variable>)"</code> .

<code>break.time.by, xlim</code>	numeric time-axis break interval and axis limits, applied to <i>both</i> the curves and the risk table so the two panels line up. Defaults NULL use the automatic breaks and the data's time range.
<code>tables.y.text</code>	logical. Default TRUE. If FALSE, the risk-table strata tick labels are hidden.
<code>tables.theme, fontsize</code>	the theme and font size of the risk table, as in <code>ggsurvplot()</code> .
<code>show.hr</code>	logical. Default FALSE. If TRUE, annotate the plot with the covariate hazard ratio and 95% confidence interval for the grouping variable, taken from the Cox model fit – i.e. the hazard ratio <i>adjusted</i> for the model's other terms. Drawn only when <code>variable</code> is an estimated model coefficient (not a <code>strata()</code> term, an interaction, or a non-treatment contrast such as an ordered factor); a factor with more than two levels shows one line per non-reference level.
<code>hr.coord</code>	numeric vector <code>c(x, y)</code> giving a time and survival point at which to place the hazard-ratio annotation <i>inside</i> the panel, in a translucent box. This is manual placement: on a small figure a label near the panel centre can run past an edge, so adjust the coordinate or the figure size if it does. Default NULL draws the annotation as a subtitle above the plot instead, which keeps it clear of the curves at the usual figure sizes and for any fun transform.
<code>hr.size</code>	the font size of the hazard-ratio annotation. Default 4.
<code>...</code>	further arguments passed to the function <code>ggpar</code> 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` (a `ggplot`); when `risk.table` is not `FALSE`, returns a [ggsurvplot](#) object (curves + aligned number-at-risk table) instead.

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")

# With a Kaplan-Meier number-at-risk table by the grouping variable
#::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
ggadjustedcurves(fit2, data = bladder, method = "average", variable = "rx",
  risk.table = TRUE)

## 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. It also accepts a `cuminc` object from the **tidycmprsk** package (which carries proper pointwise confidence limits and the data), can annotate Gray's test, and can overlay the naive complement-of-Kaplan-Meier for teaching.

Usage

```

ggcompetingrisks(
  fit,
  gnames = NULL,
  gsep = " ",
  multiple_panels = TRUE,
  ggtheme = theme_survminer(),
  coef = 1.96,
  conf.int = FALSE,
  pval = FALSE,
  add.naive.km = FALSE,
  cause = NULL,
  risk.table = FALSE,
  risk.table.height = 0.25,
  risk.table.title = "Number at risk",
  risk.table.fontsize = 4.5,
  risk.table.y.text = TRUE,

```

```

    break.time.by = NULL,
    ...
  )

```

Arguments

<code>fit</code>	a cumulative-incidence object: a <code>cmprsk::cuminc</code> , a <code>cuminc</code> (<code>tidycmprsk</code>), or a <code>survfitms</code> from <code>survfit</code> .
<code>gnames</code>	a vector with group names. If not supplied then will be extracted from <code>fit</code> object (<code>cuminc</code> only).
<code>gsep</code>	a separator that extracts group names and event names from <code>gnames</code> object (<code>cuminc</code> only).
<code>multiple_panels</code>	if <code>TRUE</code> then groups will be plotted in different panels (<code>cuminc</code> only).
<code>ggtheme</code>	function, <code>ggplot2</code> theme name. Default value is <code>theme_survminer</code> . Allowed values include <code>ggplot2</code> official themes: see <code>theme</code> .
<code>coef</code>	see <code>conf.int</code> , scaling factor for the ribbon. The default value is 1.96.
<code>conf.int</code>	if <code>TRUE</code> then additional layer (<code>geom_ribbon</code>) is added around the point estimate. The ribbon is plotted with boundaries $\pm$ <code>coef</code> *standard deviation.
<code>pval</code>	logical or a character string. If <code>TRUE</code> , annotate Gray's test p-value for each cause (comparing the groups) as a subtitle; a character string is used verbatim as the subtitle. Gray's test is available only for a <code>cmprsk</code> / <code>tidycmprsk</code> <code>cuminc</code> object with more than one group; for other inputs <code>pval</code> is ignored with a message (Gray, 1988).
<code>add.naive.km</code>	logical. If <code>TRUE</code> , overlay the naive complement-of-Kaplan-Meier for one cause (a dashed line), which censors the competing events and therefore <i>overestimates</i> the cumulative incidence – the classic teaching contrast (Putter, Fiocco and Geskus, 2007). Supported for a <code>cuminc</code> object (which carries the data) on the faceted layout; ignored with a message otherwise. It is therefore not drawn together with <code>risk.table = TRUE</code> , which uses the overlaid single-panel layout.
<code>cause</code>	the cause to overlay when <code>add.naive.km = TRUE</code> . Required when the model has more than one cause.
<code>risk.table</code>	logical or a character string. If <code>TRUE</code> , draw a number-at-risk table beneath the cumulative-incidence curves, locked to the same time axis. A character string selects the table content, one of "absolute" (the default for <code>TRUE</code>), "percentage", "abs_pct", "nrisk_cumcensor" or "nrisk_cumevents" (as in <code>ggsurvplot</code>). The table is available for a <code>cuminc</code> object (which carries the data), on the overlaid single-panel layout; it is drawn with <code>multiple_panels = FALSE</code> (a message is shown if <code>multiple_panels = TRUE</code> was passed explicitly). For a <code>cmprsk::cuminc</code> or <code>survfitms</code> input the at-risk counts are not available in a single overlaid panel, so <code>risk.table</code> is ignored with a message. The number at risk is the count still event-free and under observation (competing events count as leaving the risk set), matching the Kaplan-Meier at-risk convention of <code>ggsurvplot</code> .
<code>risk.table.height</code>	the height of the risk table as a fraction of the whole figure. Default 0.25.

`risk.table.title`
the title of the risk table. Default "Number at risk".

`risk.table.fontsize`
the font size of the risk-table numbers. Default 4.5.

`risk.table.y.text`
logical. If TRUE (default) the risk-table rows are labelled by group; if FALSE a short dash is shown instead.

`break.time.by`
numeric value controlling the time-axis breaks shared by the curves and the risk table. Default NULL uses automatic breaks.

...
further arguments passed to the function `ggpar` for customizing the plot.

Value

Returns a gg object; with `risk.table = TRUE` a compound `ggcompetingrisks` object (a list with `$plot` and `$table`) that prints the curves above the aligned table and works with `ggsave()`. The cumulative-incidence data and, where computed, Gray's test table and the naive-KM data are attached as `attr(x, "cr.data")`, `attr(x, "grays.test")` and `attr(x, "naive.km")`.

Author(s)

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

References

Gray, R. J. (1988). A class of K-sample tests for comparing the cumulative incidence of a competing risk. *The Annals of Statistics*, 16(3), 1141-1154. doi:10.1214/aos/1176350951.

Putter, H., Fiocco, M. and Geskus, R. B. (2007). Tutorial in biostatistics: competing risks and multi-state models. *Statistics in Medicine*, 26(11), 2389-2430. doi:10.1002/sim.2712.

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)

# tidycmprsk input: Gray's test p-value and the naive-KM teaching overlay
if (require("tidycmprsk")) {
  dd <- data.frame(time = ss, group = gg, status = cc)
  tfit <- tidycmprsk::cuminc(Surv(time, status) ~ group, data = dd)
  ggcompetingrisks(tfit, pval = TRUE)
  ggcompetingrisks(tfit, add.naive.km = TRUE, cause = "death")
}
}

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

<code>fit</code>	an object of class coxph.object - created with coxph function.
<code>type</code>	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.
<code>...</code>	further arguments passed to residuals.coxph or to the function ggpar for customizing the plot.
<code>linear.predictions</code>	(deprecated, see <code>ox.scale</code>) a logical value indicating whether to show linear predictions for observations (TRUE) or just indexed of observations (FALSE) on X axis.
<code>ox.scale</code>	one value from <code>c("linear.predictions", "observation.id", "time")</code> . It defines what will be presented on OX scale. Possible values: <code>y hat</code> for "linear.predictions", Id of an observation for "observation.id" or Time for "time".
<code>hline</code>	a logical - should the horizontal line be added to highlight the $Y=0$ level.
<code>sline, sline.se</code>	a logical - should the smooth line be added to highlight the local average for residuals.
<code>hline.col, hline.size, hline.lty, hline.alpha, hline.yintercept</code>	color, size, linetype, visibility and Y-axis coordinate to be used for geom_hline . Used only when <code>hline = TRUE</code> .
<code>sline.col, sline.size, sline.lty, sline.alpha</code>	color, size, linetype and visibility to be used for geom_smooth . Used only when <code>sline = TRUE</code> .
<code>point.col, point.size, point.shape, point.alpha</code>	color, size, shape and visibility to be used for points.
<code>title, subtitle, caption</code>	main title, subtitle and caption.
<code>ggtheme</code>	function, ggplot2 theme name. Default value is <code>ggplot2::theme_bw()</code> . 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. An alternative to fit when you have not fitted a coxph model; supply data as well.
data	a data.frame in which to interpret the variables named in the formula,
fit	an object of class coxph.object - created with the coxph function. Provide either fit, or formula and data; the diagnostic is identical (it uses only the model formula).
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")

# equivalently, from a formula + data (no fitted model needed)
ggcoxfunctional(Surv(futime, death) ~ mspike + log(mspike) + age,
  data = mgus)
```

ggcoxnp

Non-Proportional-Hazards Diagnostic Panel

Description

Draws a small-multiple of complementary diagnostics for the proportional-hazards (PH) assumption of a Cox model, for one covariate of interest, and – when PH does not hold – shows the *shape* of the time-varying effect. Up to four panels are produced:

1. **cloglog** – log cumulative hazard $\log(-\log S(t))$ against log time, by group. Parallel curves are consistent with PH; converging or crossing curves indicate non-proportional hazards.
2. **schoenfeld** – the scaled Schoenfeld residuals for the covariate with a natural-spline smooth and a pointwise band, an estimate of the time-varying coefficient $\beta(t)$ (Grambsch and Therneau, 1994). A curve flat at $\hat{\beta}$ is consistent with PH; the Grambsch-Therneau test p-value is shown.
3. **hr** – the hazard-ratio *trend* $\exp(\beta(t))$ from the same smooth, on a log axis, with the constant Cox hazard ratio $\exp(\hat{\beta})$ and $HR = 1$ for reference. This shows the *direction and shape* of a time-varying effect; it is a first-order residual-based diagnostic and *understates the magnitude* of strong departures – for a quantitative time-varying hazard ratio fit a model with `tt()` or a piecewise/interaction term.
4. **rmst** – the difference in restricted mean survival time between two groups as a function of the horizon τ , with a pointwise band. A band that stays away from zero as τ grows indicates a durable benefit; one that shrinks toward zero indicates an effect confined to earlier times.

Panels `cloglog` and `rmst` are *marginal* (unadjusted, by group); `schoenfeld` and `hr` are the covariate's effect *adjusted* for the model's other terms. They answer complementary questions and can disagree, which is itself informative.

Usage

```
ggcoxnp(
  fit,
  variable = NULL,
  data = NULL,
  panels = c("cloglog", "schoenfeld", "hr", "rmst"),
  transform = "km",
  tau = NULL,
  conf.level = 0.95,
  df = 4,
  numeric.split = FALSE,
  palette = NULL,
  ggtheme = theme_survminer(),
  title = NULL,
  ...
)

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

## S3 method for class 'ggcoxnp'
grid.draw(x, recording = TRUE)
```

Arguments

<code>fit</code>	a coxph model object.
<code>variable</code>	name of the covariate to diagnose. Required when the model has more than one term; if the model has a single term it is used by default.
<code>data</code>	the data frame used to fit <code>fit</code> . Only needed when the model data cannot be reconstructed from <code>fit</code> automatically.
<code>panels</code>	which panels to draw, any of "cloglog", "schoenfeld", "hr", "rmst". Panels that do not apply to the chosen covariate are dropped with a message (e.g. hr needs a single-coefficient term; cloglog/rmst need groups).
<code>transform</code>	time transform passed to cox.zph for the schoenfeld/hr panels. One of "km" (default), "rank", "identity", "log". The smooth is always drawn on the real-time axis; the transform affects how event times are spaced for the fit and can change the drawn shape.
<code>tau</code>	maximum horizon for the rmst panel. NULL (default) uses the largest admissible time.
<code>conf.level</code>	confidence level for the pointwise bands, a single number in (0, 1). Default 0.95.
<code>df</code>	degrees of freedom of the natural-spline smooth of the Schoenfeld residuals. Default 4; <code>df = 2</code> gives an almost-linear fit.
<code>numeric.split</code>	for a numeric covariate, whether to split it at its median to form the two groups used by the cloglog/rmst panels. Default FALSE (those panels are skipped for a numeric covariate).

<code>palette</code>	the colour palette for the group panels, passed to <code>ggpar</code> (e.g. a name like "jco" or a colour vector).
<code>ggtheme</code>	a <code>ggplot2</code> theme. Default <code>theme_survminer()</code> .
<code>title</code>	an overall title used when the panel is printed.
<code>...</code>	additional arguments (currently unused). Passing 'conf.int' here is an error: use 'conf.level'.
<code>x</code>	a <code>ggcoxnp</code> object.
<code>newpage</code>	logical; whether to draw on a fresh graphics page.
<code>recording</code>	logical; passed by <code>grid.draw</code> (unused here).

Value

an object of class `c("ggcoxnp", "ggsurv", "list")`: a named list of the drawn `ggplots` (one per surviving panel). Printing it arranges the panels into a grid. The computed data behind the panels are attached as `attr(x, "data")` (a list with, where applicable, `beta_t` and `rmst_tau` data frames).

References

- Grambsch, P. M. and Therneau, T. M. (1994). Proportional hazards tests and diagnostics based on weighted residuals. *Biometrika*, 81(3), 515-526. doi:[10.1093/biomet/81.3.515](https://doi.org/10.1093/biomet/81.3.515).
- Royston, P. and Parmar, M. K. B. (2013). Restricted mean survival time: an alternative to the hazard ratio for the design and analysis of randomized trials with a time-to-event outcome. *BMC Medical Research Methodology*, 13, 152. doi:[10.1186/1471228813152](https://doi.org/10.1186/1471228813152).

See Also

[ggcoxzph](#) for the standalone scaled-Schoenfeld plot (one panel per coefficient), [ggcoxdiagnostics](#), [ggrmst](#).

Examples

```
library(survival)

# Two-arm treatment: all four panels
lung2 <- lung
lung2$sex <- factor(lung2$sex, labels = c("Male", "Female"))
fit <- coxph(Surv(time, status) ~ sex + age, data = lung2)
ggcoxnp(fit, variable = "sex")

# A single covariate is used by default
fit1 <- coxph(Surv(time, status) ~ sex, data = lung2)
ggcoxnp(fit1)
```

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,
  var_pval = NULL,
  add.beta.line = FALSE,
  beta.line.col = "blue",
  beta.line.type = "dashed",
  beta.line.size = 0.5,
  ggtheme = theme_survminer(),
  ...
)

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

Arguments

<code>fit</code>	an object of class cox.zph , or a coxph model, in which case cox.zph() is run on it automatically.
<code>resid</code>	a logical value, if TRUE the residuals are included on the plot, as well as the smooth fit.
<code>se</code>	a logical value, if TRUE, confidence bands at two standard errors will be added.
<code>df</code>	the degrees of freedom for the fitted natural spline, <code>df=2</code> leads to a linear fit.
<code>nsmo</code>	number of points used to plot the fitted spline.
<code>var</code>	the set of variables for which plots are desired. By default, plots are produced in turn for each variable of a model.

<code>point.col</code> , <code>point.size</code> , <code>point.shape</code> , <code>point.alpha</code>	color, size, shape and visibility to be used for points.
<code>caption</code>	the caption of the final grob (bottom in arrangeGrob)
<code>var_pval</code>	optional numeric threshold; an alternative to <code>var</code> that selects only the terms whose Grambsch-Therneau proportional-hazards test p-value is below this value (e.g. <code>var_pval = 0.05</code> to plot only terms flagged for non-proportionality). Cannot be used together with <code>var</code> . Default NULL (all terms).
<code>add.beta.line</code>	logical; if TRUE, overlay a horizontal reference line at the model's estimated coefficient for each term (under proportional hazards the scaled Schoenfeld residuals scatter around β). Requires a <code>coxph</code> model as input (a <code>cox.zph</code> object does not carry the coefficients). The line is drawn only for single-coefficient terms; a multi-level factor term has one panel but several coefficients, so no single reference line applies and it is left without one. Default FALSE.
<code>beta.line.col</code> , <code>beta.line.type</code> , <code>beta.line.size</code>	color, line type and line width of the beta reference line (used when <code>add.beta.line = TRUE</code>).
<code>ggtheme</code>	function, ggplot2 theme name. Allowed values include ggplot2 official themes: see theme .
<code>...</code>	further arguments passed to either the <code>print()</code> function or to the ggpar function for customizing the plot (see Details section).
<code>x</code>	an object of class <code>ggcoxzph</code>
<code>newpage</code>	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

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

<code>conf.int.km</code>	same as <code>conf.in.flex</code> but for the kaplan-meier estimates.
<code>legend.labs</code>	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.
<code>...</code>	additional arguments passed to the function <code>ggsurvplot()</code> .

Value

a `ggsurvplot`

Author(s)

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

See Also

[ggsurvparametric](#), which draws the same parametric-over-KM overlay for `survreg` models as well as `flexsurvreg`.

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,
  global.stats = TRUE,
  ref.display = TRUE,
```

```

var.labels = NULL,
palette = NULL,
columns = c("N", "hr", "ci", "p"),
ggtheme = NULL
)

```

Arguments

<code>model</code>	an object of class <code>coxph</code> .
<code>data</code>	a dataset used to fit survival curves. If not supplied then data will be extracted from 'fit' object.
<code>main</code>	title of the plot.
<code>cpositions</code>	relative positions of first three columns in the OX scale.
<code>fontsize</code>	relative size of annotations in the plot. Default value: 0.7.
<code>refLabel</code>	label for reference levels of factor variables. Either a single string (default "reference", applied to every reference level) or a named character vector mapping a variable name to its reference label, e.g. <code>c(sex = "female (ref)", rx = "Obs (ref)")</code> ; variables not named keep the default label. Match is on the model term name (before any <code>var.labels</code> relabelling).
<code>noDigits</code>	number of digits for estimates and p-values in the plot.
<code>global.stats</code>	logical value. Default is TRUE. If FALSE, the bottom caption reporting the global statistics (number of events, global likelihood-ratio-test p-value, AIC and concordance index) is omitted. Useful when arranging several forest plots in a panel.
<code>ref.display</code>	logical value. Default is TRUE. If FALSE, the rows that have no hazard ratio – factor baselines, and any non-estimable / aliased or <code>strata()</code> terms, i.e. the rows labelled <code>refLabel</code> ("reference") – are omitted from both the plot and the table, keeping only the estimated levels. Default TRUE shows every row (unchanged).
<code>var.labels</code>	a named character vector giving display labels for the variable names, e.g. <code>c(age = "Age (years)", sex = "Sex")</code> . The names must match the model term names (as shown in the plot by default); only matched terms are relabelled, unmatched terms keep their original name. Default NULL uses the term names unchanged.
<code>palette</code>	the colour of the hazard-ratio points and confidence intervals. Default NULL draws them in black (unchanged). A single colour (name or hex) draws every point/interval in that one accent colour – the recommended, tasteful choice, since colour on a forest plot does not otherwise carry meaning. Alternatively a palette – a vector of colours or a palette name accepted by get_palette (e.g. "npg", "lancet") – colours the points by variable (one colour per variable); no colour legend is drawn, as the rows are already labelled. A colour vector longer than the number of variables is interpolated (<code>get_palette</code> behaviour). Prefer a palette without grey (e.g. "npg") so a point is not low-contrast on the shaded rows.
<code>columns</code>	character vector selecting which statistic columns to display, any subset of <code>c("N", "hr", "ci", "p")</code> – the sample size, the hazard-ratio value, the confidence interval, and the p-value / significance stars. The variable name, the level and the

forest itself are always shown. Case-insensitive; order is ignored (the column positions are fixed). A dropped column reflows so a stacked partner re-centres and the p-value's reserved right margin is reclaimed; the default (all four) is unchanged. Note the reference-level label (`refLabel`) is drawn in the "hr" column, so omitting "hr" also hides it.

`ggtheme` a `ggplot2` theme (e.g. the result of `theme(...)` or a complete theme) applied to the forest plot. Because `ggforest()` returns a rasterised `gtable`, a theme added to the returned object has no effect; pass it here instead, e.g. `ggtheme = theme(text = element_text(size = 14))`. It is added after the built-in theme: a partial theme `(...)` overrides only the matching elements, whereas a complete theme (e.g. `theme_void()`) replaces the built-in look entirely. Default `NULL` leaves the appearance unchanged.

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)

# a single accent colour, and per-variable reference labels
ggforest(bigmodel, palette = "#0073C2",
          refLabel = c(sex = "female (ref)", rx = "Obs (ref)"))

# a clean, publication-ready look: one accent colour and a slightly larger
# font. Add global.stats = FALSE when placing several forests in one figure.
ggforest(bigmodel, palette = "#0073C2", fontsize = 0.9)
```

Description

Draws a forest plot that compares the hazard ratio of a single covariate across several `coxph` models – one row per model. It is the tool for the crude-versus-adjusted (or naive-versus-time-varying) comparison: how a covariate's effect changes as the model specification changes.

It is the model-wise sibling of `ggforest` (rows are the terms of one model) and `ggforest_subgroup` (rows are subgroups).

Usage

```
ggforest_models(
  models,
  term,
  conf.level = 0.95,
  model.names = NULL,
  show.p = TRUE,
  show.n = TRUE,
  favours = NULL,
  point.size.by.precision = FALSE,
  main = "Hazard ratio comparison",
  xlab = NULL,
  noDigits = 2,
  ggtheme = theme_survminer()
)
```

Arguments

<code>models</code>	a named list of <code>coxph</code> models. The list names are used as the row labels (falling back to <code>model.names</code> , then "Model 1", "Model 2", ...).
<code>term</code>	the model variable (or a single coefficient name) whose hazard ratio is compared across the models.
<code>conf.level</code>	the confidence level for the intervals, a single number in (0, 1). Default 0.95.
<code>model.names</code>	optional character vector of row labels, used when the <code>models</code> list is unnamed.
<code>show.p</code>	logical. Show the per-model Wald p-value column. Default TRUE.
<code>show.n</code>	logical. Show the events / N column (models fitted on different data have different sizes). Default TRUE.
<code>favours</code>	optional length-2 character vector, e.g. <code>c("Favours treatment", "Favours control")</code> , annotated under the reference line.
<code>point.size.by.precision</code>	logical. Scale the point size by the estimate's precision (1/se). Default FALSE – with few model rows a size difference is easily misread (a more-adjusted model has a larger standard error, hence a smaller box).

<code>main</code>	plot title. Default "Hazard ratio comparison".
<code>xlab</code>	x-axis label. Default NULL builds one from term and the confidence level; for a factor coefficient (e.g. "sexFemale") pass a friendlier label such as "Hazard ratio (female vs male)".
<code>noDigits</code>	number of digits for the hazard ratio and CI. Default 2.
<code>ggtheme</code>	a ggplot2 theme. Default <code>theme_survminer()</code> .

Details

The hazard ratio, confidence interval and Wald p-value for term are read from each model with base **survival** (`coef()` / `vcov()`), so a robust (`cluster()`) variance is used when present. term is a model *variable* name (e.g. "age", "sex"); for a factor with more than one non-reference level it must resolve to a single coefficient, otherwise an error lists the coefficient names to choose from (e.g. "rxLev"). A model that does not contain term is dropped with a warning.

Because the models may be fitted on different data or with different specifications, one model's very wide confidence interval could otherwise crush the shared axis and make the others incomparable. The interval whiskers are therefore clamped to a robust window around the point estimates, with an arrow drawn where a whisker is clipped; the full numeric interval always appears in the "HR (CI)" column, so nothing is hidden.

Value

a ggplot (an assembled multi-panel figure).

See Also

[ggforest](#), [ggforest_subgroup](#).

Examples

```
library(survival)
m.crude    <- coxph(Surv(time, status) ~ age, data = lung)
m.adjusted <- coxph(Surv(time, status) ~ age + sex + ph.ecog, data = lung)
ggforest_models(list("Crude" = m.crude, "Adjusted" = m.adjusted), term = "age")
```

ggforest_subgroup

Subgroup Forest Plot of a Treatment Hazard Ratio

Description

Draws the forest plot clinicians publish for a subgroup analysis: the hazard ratio of a treatment *within* each level of one or more subgroup variables, together with the treatment-by-subgroup interaction test. It complements [ggforest](#)(), which draws the coefficients of a single fitted model rather than a treatment effect broken down by subgroup.

For each subgroup level the Cox model is refit on that subset and the treatment hazard ratio (with confidence interval) is extracted. For each subgroup *variable* a single interaction p-value is reported,

from a likelihood-ratio test of the treatment-by-subgroup interaction – this test, not the per-level hazard ratios, is the evidence for effect modification.

Usage

```
ggforest_subgroup(
  model,
  data = NULL,
  treatment,
  subgroups,
  conf.level = 0.95,
  show.overall = TRUE,
  show.pinteraction = TRUE,
  show.n = TRUE,
  favours = NULL,
  point.size.by.precision = TRUE,
  main = "Treatment effect by subgroup",
  xlab = NULL,
  noDigits = 2,
  ggtheme = theme_survminer()
)
```

Arguments

<code>model</code>	a coxph object whose terms include treatment. It may be crude (<code>~ treatment</code>) or adjusted (<code>~ treatment + covariates</code>).
<code>data</code>	the data frame used to fit model. If not supplied it is extracted from model.
<code>treatment</code>	the name of the treatment variable (a term of model). It must yield a single hazard ratio, i.e. a two-level factor or a numeric variable; a treatment with more than two levels is ambiguous here and is an error.
<code>subgroups</code>	subgroup variable names in data (categorical / factor). May be named, in which case the names are used as the display labels, e.g. <code>c(Sex = "sex", "Age group" = "age.grp")</code> . Continuous variables must be binned first.
<code>conf.level</code>	the confidence level for the intervals, a single number in (0, 1). Default 0.95.
<code>show.overall</code>	logical; add an overall (all-subjects) treatment hazard ratio row at the top. Default TRUE.
<code>show.pinteraction</code>	logical; show the per-variable interaction p-value. Default TRUE.
<code>show.n</code>	logical; show a "No. of patients (%)" column giving the number of subjects in each subgroup level (and their percentage of the total). Default TRUE. This count is the subgroup size; when the model has adjusting covariates with missing values it can exceed the complete-case sample the hazard ratio is actually fit on.
<code>favours</code>	optional length-2 character vector <code>c("Favours treatment", "Favours control")</code> drawn under the axis on the left ($HR < 1$) and right ($HR > 1$) sides of the reference line. Default NULL omits it.

<code>point.size.by.precision</code>	logical; scale the hazard-ratio box area by the estimate's precision (inverse variance), the usual forest-plot convention, so more informative subgroups draw a larger box. Default TRUE; set FALSE for equal-sized points.
<code>main</code>	plot title. Default "Treatment effect by subgroup".
<code>xlab</code>	x-axis label. Default NULL builds "Hazard ratio (<treatment> vs <reference>, log scale)".
<code>noDigits</code>	number of digits for the hazard ratios and p-values. Default 2.
<code>ggtheme</code>	a ggplot2 theme for the forest panel. Default <code>theme_survminer()</code> .

Details

The per-level estimate is the treatment coefficient of model refit on the subset of data in that level (any subgroup variable that also appears in the model, and any adjusting covariate that becomes constant within the subset, is dropped from that fit so it does not error). Levels in which the model cannot be fit – too few events, no treatment contrast, or non-convergence – are dropped from the plot with a warning rather than stopping.

The interaction p-value for a subgroup variable is the likelihood-ratio test comparing the additive model (model plus the subgroup main effect) with the interaction model (additionally the treatment-by-subgroup term), fit on the observations complete for both. As with any subgroup analysis the interaction test is the inference; the individual per-level hazard ratios are descriptive and subject to multiplicity (see Wang et al., 2007).

The plot is composed of three aligned panels (labels, forest, statistics), so the text columns keep a fixed width and do not collide with the forest at any figure size.

Value

a ggplot object (the assembled panels).

References

Wang R, Lagakos SW, Ware JH, Hunter DJ, Drazen JM (2007). Statistics in medicine – reporting of subgroup analyses in clinical trials. *New England Journal of Medicine* 357(21):2189-2194.

See Also

[ggforest\(\)](#)

Examples

```
library(survival)
# Two treatment arms (Lev+5FU vs Obs) with categorical subgroups
cc <- colon[colon$type == 2 & colon$rx %in% c("Obs", "Lev+5FU"), ]
cc$rx <- droplevels(cc$rx)
cc$sex <- factor(cc$sex, labels = c("Female", "Male"))
cc$age.grp <- factor(ifelse(cc$age >= 60, ">=60", "<60"), levels = c("<60", ">=60"))
cc$differ <- factor(cc$differ, labels = c("well", "moderate", "poor"))

fit <- coxph(Surv(time, status) ~ rx, data = cc)
```

```
ggforest_subgroup(fit, data = cc, treatment = "rx",
  subgroups = c(Sex = "sex", Age = "age.grp", Differentiation = "differ"))
```

gglandmark

Landmark Analysis of Survival Curves

Description

Draws Kaplan-Meier curves **re-originated at a landmark time L**: the analysis is restricted to subjects still event-free and under observation at L, the time axis is reset so that L becomes time 0, and the curves are re-estimated from the landmark onward.

Landmark analysis is the standard remedy for **immortal-time (guarantee-time) bias** that arises when groups are defined by a post-baseline status (e.g. tumour response, transplantation): comparing such groups from study baseline credits the responders with the event-free time they were guaranteed to accrue *before* they could be classified. Conditioning on being event-free at a fixed landmark removes that bias (Anderson, Cain & Gelber, 1983; Dafni, 2011).

Usage

```
gglandmark(
  fit,
  data = NULL,
  landmark.time,
  conf.int = FALSE,
  xlab = "Time since landmark",
  ...
)
```

Arguments

<code>fit</code>	a survfit object (<code>survival::survfit()</code> or <code>survminer's surv_fit()</code>).
<code>data</code>	the data frame used to fit the curves. Strongly recommended: it is required to rebuild the landmark cohort and can only be recovered automatically for a fit created in the global environment (see ggsurvplot).
<code>landmark.time</code>	the landmark time L (in the data's time units) at which the clock is reset. Required.
<code>conf.int</code>	logical; draw the confidence-interval band of the re-originated curves. Default FALSE (matching ggsurvplot()). Because the curves are re-fitted on the landmark cohort, the band is drawn at <code>survfit()</code> 's default 95% level rather than the level fit was built with.
<code>xlab</code>	x-axis label. Default "Time since landmark" to make the re-originating explicit.
<code>...</code>	other arguments passed to ggsurvplot() (e.g. <code>palette</code> , <code>legend.labs</code> , <code>pval</code> , <code>risk.table</code>). A subtitle supplied here is respected; otherwise a subtitle naming the landmark and cohort size is added.

Details

The landmark cohort is every subject with `time >= landmark.time` (still at risk at L). Each retained subject's clock is reset to `time - landmark.time` with its event status unchanged, and the Kaplan-Meier curves are re-fitted on this re-originated data via `surv_fit()` and drawn with `ggsurvplot()`. Because the cohort is event-free at L, every curve correctly starts at $S = 1$; a subject whose *event* falls exactly on L has, by definition, already had the event by the landmark and is excluded (reported in a message and the subtitle) so the re-originated curve is not pulled below 1 at time 0.

Any log-rank p-value or table you request through ... (e.g. `pval = TRUE`, `risk.table = TRUE`) is computed on the re-originated landmark cohort, as it must be for a valid landmark comparison.

Caveats. Landmark analysis discards subjects with an event or censoring before L (a loss of power) and is sensitive to the – necessarily arbitrary – choice of L; reporting a sensitivity analysis over several landmarks is good practice (Dafni, 2011). The immortal-time-bias correction applies when the grouping variable reflects a status attained up to L; for a fixed baseline covariate there is no immortal time to remove.

Groups are taken from the fit's own strata, so the labels and their order match `names(fit$strata)`, and a transformed right-hand side such as `~ (age > 60)` is honoured as written (an ungrouped `~ 1` fit has no strata and is treated as a single group). Note that the landmark curves are a *refit* on the landmark cohort, so a data-dependent grouping term such as `cut(age, 3)` recomputes its breaks on the survivors; bin such a variable before fitting if the strata must match the original figure exactly. A weighted `survfit()` is refused, since the landmark cohort would have to be re-fitted and the confidence-interval convention for weighted data is ambiguous.

Value

a `ggsurvplot` object (the re-originated curves, plus any requested risk table). The landmark details – `landmark.time`, the number at risk, the number of events dropped at the landmark, and the re-originated fit – are attached as `attr(x$plot, "landmark")`.

References

- Anderson JR, Cain KC, Gelber RD (1983). Analysis of survival by tumor response. *Journal of Clinical Oncology* 1(11):710-719.
- Dafni U (2011). Landmark analysis at the 25-year landmark point. *Circulation: Cardiovascular Quality and Outcomes* 4(3):363-371.

See Also

[ggmilestone](#), [ggsurvplot](#), [ggrmst](#).

Examples

```
library(survival)
fit <- survfit(Surv(time, status) ~ sex, data = lung)

# Re-originate the curves at day 200 (only day-200 survivors, clock reset)
gglandmark(fit, data = lung, landmark.time = 200)

# With a log-rank p-value and risk table on the landmark cohort
```

```
gglandmark(fit, data = lung, landmark.time = 200,
            pval = TRUE, risk.table = TRUE, palette = "jco")
```

ggmilestone

Milestone (Fixed-Time) Survival

Description

Annotates Kaplan-Meier curves with **milestone survival**: the survival probability $S(t)$ at one or more clinically meaningful fixed times (e.g. 1- and 2-year survival) with its confidence interval, and – for a two-arm comparison – the **between-arm difference** in milestone survival with its confidence interval and p-value. Milestone survival is a standard trial endpoint when a single hazard ratio is not enough (e.g. under non-proportional hazards).

Usage

```
ggmilestone(
  fit,
  data = NULL,
  milestone.times,
  conf.int = FALSE,
  conf.level = 0.95,
  ref.group = NULL,
  label.milestones = NULL,
  ...
)
```

Arguments

<code>fit</code>	a survfit object (<code>survival::survfit()</code> or <code>survminer's surv_fit()</code>).
<code>data</code>	the data frame used to fit the curves (see ggsurvplot).
<code>milestone.times</code>	numeric vector of the fixed times at which to read survival (in the data's time units). Required.
<code>conf.int</code>	logical; draw the confidence-interval band of the Kaplan-Meier curves. Default FALSE (matching ggsurvplot()). The band is drawn at the confidence level <code>fit</code> was built with; <code>conf.level</code> sets the level of the milestone estimates and their difference, not of the band.
<code>conf.level</code>	confidence level for the milestone survival intervals and the between-arm difference. Default 0.95.
<code>ref.group</code>	for three or more arms, the reference arm the milestone differences are taken against, given in the fit's own label form (e.g. "sex=1"; the error message lists the valid values). Default NULL reports only the per-arm milestone survival. Ignored for one or two arms.

`label.milestones` logical or NULL; draw the per-arm $S(t)$ value labels on the curves. Default NULL labels them only when the plot stays legible (arms x milestones ≤ 4) and otherwise omits them (the values remain in the caption and the attached table). TRUE/FALSE force it on/off.

... other arguments passed to `ggsurvplot()` (e.g. `palette`, `legend.labs`, `risk.table`). A caption supplied here is respected; otherwise the milestone summary is placed there.

Details

For each arm and milestone the estimate is the Kaplan-Meier $S(t)$ with its Greenwood standard error and confidence interval (`summary.survfit`; the interval uses the same `conf.type` as `fit`). The between-arm difference is on the risk-difference (identity) scale – the second arm minus the first – with $se = \sqrt{se_A^2 + se_B^2}$ (arms independent), a normal confidence interval and Wald p-value – the quantity clinical reports quote. Note (Klein et al., 2007) that this naive identity-scale inference can have inflated type-I error and poor coverage when S is close to 0 or 1 or the sample is small; interpret such cases with care.

A milestone time beyond an arm's follow-up is **not** estimable unless every subject in that arm has already had the event (the curve has reached $S = 0$): otherwise that arm's $S(t)$ is returned as NA with a warning, rather than silently dropped or carried forward as `summary.survfit` would.

The between-arm difference is defined only for two arms; for three or more arms, set `ref.group` to report each other arm's difference against a reference (otherwise only the per-arm milestone table is produced). For a single arm only the per-arm milestone survival is shown.

Arms are taken from the fit's own strata, so the labels and their order match `names(fit$strata)` and the curves drawn by `ggsurvplot()`, and a transformed right-hand side such as `~ (age > 60)` is honoured as written. (An ungrouped `~ 1` fit has no strata and is reported as a single "All" arm.) For two arms the difference is the *second* arm minus the first (the first level is the reference), matching `ggmst_difference()`; the row is always labelled with the contrast, so the direction is never left to the reader. A weighted `survfit()` is refused: the confidence-interval convention for weighted data is ambiguous, so reporting an interval for it would overstate what the estimate supports.

Value

a `ggsurvplot` object: the Kaplan-Meier curves with a dashed line at each milestone, the per-arm $S(t)$ marked, and the milestone survival (and between-arm difference) reported in the caption. The full milestone table – per-arm $S(t)$, CI, and any between-arm differences with CI and p-value – is attached as `attr(x$plot, "milestone.table")`. Its `strata` column carries the fit's own label – "All" for an ungrouped fit, as in `surv_median()` and `surv_summary()` – which is what the caption shows unless `legend.labs` renames the arms on the plot. A between-arm difference row is labelled with its contrast; only a difference row can carry a p.value, though it is NA when the difference is not estimable (a milestone beyond an arm's follow-up, or a zero standard error).

References

Klein JP, Logan B, Harhoff M, Andersen PK (2007). Analyzing survival curves at a fixed point in time. *Statistics in Medicine* 26(24):4505-4519.

See Also

[gglandmark](#), [ggrmst](#), [ggsurvplot](#).

Examples

```
library(survival)
fit <- survfit(Surv(time, status) ~ sex, data = lung)

# 1- and 2-year (365 / 730 day) milestone survival with the between-arm difference
ggmilestone(fit, data = lung, milestone.times = c(365, 730))

# recover the full milestone table
p <- ggmilestone(fit, data = lung, milestone.times = c(365, 730))
attr(p$plot, "milestone.table")
```

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 = "",
hjust = 0.5,
axes.offset = TRUE,
ggtheme = theme_survminer(),
tables.theme = ggtheme,
obscure.less.than = NULL,
obscure.zero = FALSE,
...
)

```

Arguments

<code>fit</code>	an object of class <code>survfit</code> . Can be a list containing two components: 1) <code>time</code> : time variable used in <code>survfit</code> ; 2) <code>table</code> : survival table as generated by the internal function <code>.get_timepoints_survsummary()</code> . Can be also a simple data frame.
<code>data</code>	a dataset used to fit survival curves. If not supplied then data will be extracted from 'fit' object.
<code>risk.table.type</code>	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 <code>survtable = "risk.table"</code> .
<code>...</code>	other arguments passed to the function ggsurvtable and ggpar .
<code>survtable</code>	a character string specifying the type of survival table to plot.
<code>title</code>	the title of the plot.

<code>risk.table.title</code>	The title to be used for the risk table.
<code>cumevents.title</code>	The title to be used for the cumulative events table.
<code>cumcensor.title</code>	The title to be used for the cumcensor table.
<code>color</code>	color to be used for the survival curves. • If the number of strata/group (<code>n.strata</code>) = 1, the expected value is the color name. For example <code>color = "blue"</code>. • If <code>n.strata > 1</code>, the expected value is the grouping variable name. By default, survival curves are colored by strata using the argument <code>color = "strata"</code>, 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 <code>palette</code>.
<code>palette</code>	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. <code>c("blue", "red")</code> ; and scientific journal palettes from ggsci R package, e.g.: "npg", "aaas", "lancet", "jco", "ucscgb", "uchicago", "simpsons" and "rickandmarty". See details section for more information. Can be also a numeric vector of <code>length(groups)</code> ; in this case a basic color palette is created using the function palette .
<code>break.time.by</code>	numeric value controlling time axis breaks. Default value is NULL.
<code>xlim</code>	x axis limits e.g. <code>xlim = c(0, 1000)</code> .
<code>xscale</code>	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 <code>c("d_m", "d_y", "m_d", "m_y", "y_d", "y_m")</code>, where d = days, m = months and y = years. For example, <code>xscale = "d_m"</code> will transform labels from days to months; <code>xscale = "m_y"</code>, will transform labels from months to years.
<code>xlab</code>	x axis label
<code>ylab</code>	y axis label
<code>xlog</code>	logical value. If TRUE, x axis is transformed into log scale.
<code>legend</code>	character specifying legend position. Allowed values are one of <code>c("top", "bottom", "left", "right", "none")</code> . Default is "top" side position. to remove the legend use <code>legend = "none"</code> . Legend position can be also specified using a numeric vector <code>c(x, y)</code> ; see details section.
<code>legend.title</code>	legend title.
<code>legend.labs</code>	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.
<code>y.text</code>	logical. Default is TRUE. If FALSE, the table y axis tick labels will be hidden.
<code>y.text.col</code>	logical. Default value is FALSE. If TRUE, the table tick labels will be colored by strata.
<code>fontsize</code>	text font size.

<code>font.family</code>	character vector specifying text element font family, e.g.: <code>font.family = "Courier New"</code> .
<code>hjust</code>	horizontal justification of the table text (passed to <code>geom_text</code>). Default is 0.5 (centered); use e.g. 0 for left-aligned.
<code>axes.offset</code>	logical value. Default is TRUE. If FALSE, set the plot axes to start at the origin.
<code>ggtheme</code>	function, ggplot2 theme name. Default value is theme_survminer . Allowed values include ggplot2 official themes: see theme .
<code>tables.theme</code>	function, ggplot2 theme name. Default value is theme_survminer . Allowed values include ggplot2 official themes: see theme . Note that, <code>tables.theme</code> is incremental to <code>ggtheme</code> .
<code>obscure.less.than</code>	optional integer for small-cell suppression (disclosure control). When set, any table count below this value is replaced by the label "<n" (e.g. <code>obscure.less.than = 5</code> shows "<5" for at-risk / event / censor counts of 1–4), so small, potentially patient-identifying counts are hidden. For the combined risk-table types (" <code>nrisk_cumevents</code> ", " <code>nrisk_cumcensor</code> ") each count in the cell is masked independently. Applies to the risk table and the cumulative event/censor tables; it does not affect the <code>ncensor.plot</code> bar chart. Default NULL applies no masking (output unchanged). Requested in survminer issue #637 .
<code>obscure.zero</code>	logical. When <code>obscure.less.than</code> is set, should a count of 0 also be masked? Default FALSE keeps zeros visible (only positive counts below the threshold are hidden).

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
```

ggrmst_difference	<i>Restricted Mean Survival Time (RMST)</i>
-------------------	---

Description

Restricted mean survival time (RMST) is the area under the Kaplan-Meier curve up to a truncation time τ – the average event-free time over $[0, \tau]$, in the original time units. It is increasingly reported alongside the hazard ratio because it stays interpretable under non-proportional hazards (ICH E9(R1)).

`ggrmst_difference()` returns a tidy table of the per-group RMST (with standard error and confidence interval) and, for a two-group comparison, the difference in RMST with its confidence interval and p-value.

`ggrmst()` draws the Kaplan-Meier curves with the RMST region shaded: for two groups the region *between* the curves up to τ is highlighted and the RMST difference annotated (that region's signed area is the difference; when the curves cross, the shaded area is the total separation and the reported difference – always the exact signed value – can be smaller); otherwise the area under each curve is shaded. A vertical line marks τ .

Usage

```
ggrmst_difference(
  fit,
  data = NULL,
  tau = NULL,
  conf.level = 0.95,
  ref.group = NULL
)
```

```
ggrmst(
  fit,
  data = NULL,
  tau = NULL,
  conf.level = 0.95,
  palette = NULL,
```

```

ggtheme = theme_survminer(),
...,
conf.int = FALSE
)

```

Arguments

<code>fit</code>	a <code>survfit</code> object (<code>survival::survfit()</code> or <code>survminer</code> 's <code>surv_fit()</code>).
<code>data</code>	the data frame used to fit the survival curves. If not supplied it is extracted from <code>fit</code> .
<code>tau</code>	numeric truncation time. Default <code>NULL</code> uses the largest time at which all groups' curves are defined (see Details). An out-of-range value is an error.
<code>conf.level</code>	the confidence level for the RMST intervals, a single number in (0, 1). Default 0.95. (<code>ggrmst_difference()</code> returns a table, so this is its only confidence argument.)
<code>ref.group</code>	for three or more groups, the reference group the RMST differences are taken against, given in the fit's own label form (e.g. "sex=1"; the error message lists the valid values). Default <code>NULL</code> reports only the per-group RMST (no differences). Ignored for one or two groups.
<code>palette, ggtheme, ...</code>	passed to <code>ggsurvplot()</code> for the underlying curves.
<code>conf.int</code>	logical; draw the confidence band around the survival curves, as in <code>ggmilestone()</code> and <code>gglandmark()</code> . Default <code>FALSE</code> . <code>ggrmst()</code> only. The band is drawn at the confidence level <code>fit</code> was built with; <code>conf.level</code> sets the level of the RMST interval, not of the band.

Details

The estimate is computed internally from the Kaplan-Meier curve (no external dependency): $RMST(\tau) = \int_0^\tau S(u) du$ by exact integration of the KM step function (held flat from the last observed time to `tau`), with the analytic variance $\sum_{t_i \leq \tau} A_i^2 d_i / (Y_i(Y_i - d_i))$ where $A_i = \int_{t_i}^\tau S(u) du$, d_i events and Y_i at risk. The difference in RMST uses the sum of the group variances and a normal confidence interval / z-test. Results match `survRM2::rmst2()` (Uno et al.); see Royston & Parmar (2013) for the methodology.

By default `tau` is the largest time at which every group's Kaplan-Meier estimate is still defined: a group whose last observation is an event places no upper limit (its curve is defined to be 0 afterwards), while a group whose last observation is censored limits `tau` to that time. A `tau` beyond that admissible range is an error.

Groups are taken from the fit's own strata, so the labels and their order match `names(fit$strata)` and the curves drawn by `ggsurvplot()`, and a transformed right-hand side such as `~ (age > 60)` is honoured as written. (An ungrouped `~ 1` fit has no strata and is reported as a single "All" group. The grouping is rebuilt from data, so a fit created with `survfit(..., subset =)` should be passed the same subset of rows.) A weighted `survfit()` is refused: the confidence-interval convention for weighted data is ambiguous, so reporting an interval for it would overstate what the estimate supports.

Value

`ggrmst_difference()` returns a data.frame with one row per group (strata, rmst, se, lower, upper, tau) and, when a two-group (or ref.group) difference is defined, additional "... - ..." rows carrying rmst (the difference), se, lower, upper and p.value. The strata column holds the fit's own labels, as `surv_median()` and `surv_summary()` do; a difference row is labelled with its contrast. Only a difference row carries a p.value; the per-group rows have NA. A comparison whose estimated variance is zero yields $p = 0$ for a non-zero difference and NaN for a zero one; treat either as uninformative rather than significant. `ggrmst()` returns a ggplot.

Functions

- `ggrmst_difference()`: a tidy table of per-group RMST and the difference.
- `ggrmst()`: the shaded Kaplan-Meier RMST plot.

References

Royston P, Parmar MKB (2013). Restricted mean survival time: an alternative to the hazard ratio for the design and analysis of randomized trials with a time-to-event outcome. *BMC Medical Research Methodology* 13:152.

Examples

```
library(survival)
fit <- survfit(Surv(time, status) ~ sex, data = lung)
ggrmst_difference(fit, data = lung)
ggrmst(fit, data = lung, palette = "jco")
```

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

<code>surv</code>	an object of Surv . If not supplied, the censoring variable is extracted from the model.
<code>fit</code>	an object of class survfit .
<code>data</code>	a dataset for predictions. If not supplied then data will be extracted from 'fit' object.
<code>type</code>	one of <code>c("cumulative", "radius", "fraction")</code> . "cumulative" stands for cumulative number of events, "radius" stands for number of events within a given radius,
<code>normalized</code>	if TRUE relative number of events is presented,
<code>censored.on.top</code>	is TRUE then censored events are on the top
<code>ggtheme</code>	function, ggplot2 theme name. Allowed values include ggplot2 official themes: see theme.
<code>palette</code>	the color palette to be used for coloring of significant variables.
<code>...</code>	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")
```

Description

Overlays the fitted survival curve(s) of a parametric model on the Kaplan-Meier estimate, so the parametric fit can be judged against the data. Accepts a [survreg](#) or a [flexsurvreg](#) model. The Kaplan-Meier is drawn as a solid step function and the fitted model as a dashed line in the same colour, one per group.

Usage

```
ggsurvparametric(
  fit,
  data = NULL,
  conf.int = FALSE,
  linewidth = 1,
  conf.level = 0.95,
  nsim = 2000,
  ggtheme = theme_survminer(),
  ...
)
```

Arguments

<code>fit</code>	a fitted parametric survival model: a survreg or flexsurvreg object.
<code>data</code>	the data frame used to fit the model. If <code>NULL</code> , it is retrieved from the model.
<code>conf.int</code>	logical. If <code>TRUE</code> , add a pointwise confidence band around the parametric curve(s). Default <code>FALSE</code> . For <code>survreg</code> the band is a parametric-bootstrap band from the model's covariance; for <code>flexsurvreg</code> it is the model's own band.
<code>linewidth</code>	width of the parametric line. Default 1.
<code>conf.level</code>	confidence level for the parametric band. Default 0.95.
<code>nsim</code>	number of draws for the <code>survreg</code> bootstrap band. Default 2000.
<code>ggtheme</code>	a <code>ggplot2</code> theme. Default theme_survminer() .
<code>...</code>	further arguments passed to ggsurvplot for the Kaplan-Meier layer (e.g. <code>palette</code> , <code>legend.labs</code> , <code>risk.table</code>). Note that <code>conf.int</code> here controls the <i>parametric</i> band, not the Kaplan-Meier band.

Details

The parametric curve is drawn only over the observed follow-up; it is not extrapolated beyond the data by default. The tail of a parametric fit is driven by the distribution assumption, so compare distributions before relying on it.

Value

the [ggsurvplot](#) object, with the parametric overlay added to its `$plot`. The computed parametric survival data is attached as `attr(x$plot, "parametric")`.

See Also

[ggflexsurvplot](#) (the flexsurv-only predecessor), [ggsurvplot](#).

Examples

```
library(survival)

# Weibull model over the Kaplan-Meier, by sex
fit <- survreg(Surv(time, status) ~ sex, data = lung, dist = "weibull")
ggsurvparametric(fit, data = lung)

# with a bootstrap confidence band
ggsurvparametric(fit, data = lung, conf.int = TRUE)
```

`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,
  output = c("classic", "ggplot"),
  preset = "none",
  ...
)

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

## S3 method for class 'ggsurvplot'
grid.draw(x, recording = 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 "rickandmarty". 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, then the computed 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. Can also be a character string ("absolute", "percentage" or "abs_pct"); "percentage" shows the count as a percent of the stratum size.
cumcensor	logical value specifying whether to show or not the table of the cumulative number of censoring. Default is FALSE. Can also be a character string ("absolute", "percentage" or "abs_pct").
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 <code>ggsurvplot_group_by()</code> 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 <code>ggsurvplot_facet()</code> function.
add.all	a logical value. If TRUE, add the survival curve of pooled patients (null model) onto the main plot. Alias of the <code>ggsurvplot_add_all()</code> function.
combine	a logical value. If TRUE, combine a list survfit objects on the same plot. Alias of the <code>ggsurvplot_combine()</code> function.
ggtheme	function, ggplot2 theme name. Default value is <code>theme_survminer</code> . Allowed values include ggplot2 official themes: see <code>theme</code> .
tables.theme	function, ggplot2 theme name. Default value is <code>theme_survminer</code> . Allowed values include ggplot2 official themes: see <code>theme</code> . Note that, <code>tables.theme</code> is incremental to <code>ggtheme</code> .
output	one of "classic" (default) or "ggplot". With "classic" the usual compound ggsurvplot object is returned (unchanged). With "ggplot" a single survival-curve ggplot is returned instead – one object that composes normally with <code>geom_*</code> , <code>ggsave()</code> and <code>facet_wrap()</code> (useful for adding your own layers, e.g. an RMST area or a cumulative-incidence overlay). Any <code>pval</code> , <code>conf.int</code> and <code>surv.median.line</code> are kept (they live on the curve). A risk / cumulative-events / cumulative-censor table or a separate censor barplot cannot be embedded in a single ggplot, so <code>risk.table</code> , <code>cumevents</code> , <code>cumcensor</code> and <code>ncensor.plot</code> are dropped with a warning if requested; use the default and its <code>\$table</code> for those. Faceting an <i>external</i> covariate (not the strata) still needs <code>ggsurvplot_facet()</code> , which recomputes the curves per panel; a plain <code>facet_wrap()</code> on the returned plot does not, and a single <code>pval</code> annotation is replicated into every panel. <code>group.by</code> and an uncombined list of fits produce several plots (not one), so <code>output = "ggplot"</code> falls back to the classic object there (with a warning).
preset	a publication preset that bundles sensible defaults so a single call yields a ready-to-publish figure. One of "none" (default, no preset), "publication" (full evidence panel: confidence bands, log-rank p-value, median-survival lines, number-at-risk table, colourblind-safe palette, clean theme), "minimal" (airy single panel, no table or bands), "classic" (grey textbook step curves with a risk table) or "presentation" (bold, large fonts for slides). A preset only sets arguments

you did **not** pass yourself, so any explicit argument (e.g. `conf.int = FALSE`, `palette = "npg"`) always overrides it. The companion themes `theme_surv_classic()`, `_minimal()` and `_bold()` can also be used on their own via `ggtheme`.

... Further 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`

`recording` kept for consistency with the `grid.draw` generic; not used.

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

an object of class `ggsurvplot` (a list) by default, or – when `output = "ggplot"` – a single survival-curve `ggplot`. The `ggsurvplot` list contains 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. May also be a vector or list of `plotmath` expressions (e.g. `c(expression(beta[1]), expression(x^2))`) to render superscripts, subscripts and Greek letters in the legend and the number-at-risk table, with the palette unchanged.

Axis limits, breaks and scales

- **break.time.by:** numeric value controlling time axis breaks. A single value gives regularly spaced breaks; a numeric vector is used as the exact break positions (e.g. `c(0, 100, 300, 600)`), keeping the curve and the risk/censor tables aligned. Default value is `NULL`.
- **break.x.by:** alias of `break.time.by`. Numeric value (or vector) 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.parse**: logical. If TRUE, a custom p-value string given via pval is parsed as a plotmath expression, allowing italic/superscript p-values (e.g. `pval = "italic(P)==1.4~x~10^-6"`). The auto-computed p-value is plain text, not plotmath. Default is FALSE.
- **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" so that weights 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 "l" (124).
- **censor.size**: numeric 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.col`, `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

```
#####
# A single composable ggplot: output = "ggplot"
#####
require("survival")
fit <- survfit(Surv(time, status) ~ sex, data = lung)
p <- ggsurvplot(fit, data = lung, conf.int = TRUE, output = "ggplot")
# p is a plain ggplot -- add your own layers, save it, facet it
```

```

p + geom_hline(yintercept = 0.5, linetype = "dashed")

#####
# 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")

```

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,  
  legend = "top",  
  ...  
)
```

Arguments

<code>fit</code>	an object of class <code>survfit</code> .
<code>data</code>	a dataset used to fit survival curves. If not supplied then data will be extracted from 'fit' object.
<code>legend.title</code>	legend title.
<code>legend.labs</code>	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.
<code>pval</code>	logical value, a numeric or a string. If logical and TRUE, the p-value is added on the plot. If numeric, then the computed 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.
<code>legend</code>	character specifying legend position. Allowed values are one of <code>c("top", "bottom", "left", "right", "none")</code> . Default is "top" side position. to remove the legend use <code>legend = "none"</code> . Legend position can be also specified using a numeric vector <code>c(x, y)</code> ; see details section.
<code>...</code>	other arguments passed to the <code>ggsurvplot()</code> 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

<code>fit</code>	an object of class <code>survfit</code> .
<code>data</code>	a dataset used to fit survival curves. If not supplied then data will be extracted from 'fit' object.
<code>fun</code>	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.
<code>surv.scale</code>	scale transformation of survival curves. Allowed values are "default" or "percent".
<code>xscale</code>	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 <code>c("d_m", "d_y", "m_d", "m_y", "y_d", "y_m")</code>, where d = days, m = months and y = years. For example, <code>xscale = "d_m"</code> will transform labels from days to months; <code>xscale = "m_y"</code>, will transform labels from months to years.
<code>color</code>	color to be used for the survival curves. • If the number of strata/group (<code>n.strata</code>) = 1, the expected value is the color name. For example <code>color = "blue"</code>. • If <code>n.strata > 1</code>, the expected value is the grouping variable name. By default, survival curves are colored by strata using the argument <code>color = "strata"</code>, 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 <code>palette</code>.

<code>palette</code>	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. <code>c("blue", "red")</code> ; and scientific journal palettes from ggsci R package, e.g.: "npg", "aaas", "lancet", "jco", "ucscgb", "uchicago", "simpsons" and "rickandmarty". 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 .
<code>linetype</code>	line types. Allowed values includes i) "strata" for changing linetypes by strata (i.e. groups); ii) a numeric vector (e.g., <code>c(1, 2)</code>) or a character vector <code>c("solid", "dashed")</code> .
<code>break.time.by</code>	numeric value controlling time axis breaks. A single value gives regularly spaced breaks (from 0 by that step); a numeric vector is used as the exact break positions (e.g. <code>c(0, 100, 300, 600)</code>), which also keeps the survival curve and the risk/censor tables aligned. Default value is NULL.
<code>break.x.by</code>	alias of <code>break.time.by</code> . Numeric value (or vector) controlling x axis breaks. Default value is NULL.
<code>break.y.by</code>	same as <code>break.x.by</code> but for y axis.
<code>conf.int</code>	logical value. If TRUE, plots confidence interval.
<code>conf.int.fill</code>	fill color to be used for confidence interval.
<code>conf.int.style</code>	confidence interval style. Allowed values include <code>c("ribbon", "step")</code> .
<code>conf.int.alpha</code>	numeric value specifying fill color transparency. Value should be in [0, 1], where 0 is full transparency and 1 is no transparency.
<code>censor</code>	logical value. If TRUE, censors will be drawn.
<code>censor.shape</code>	character or numeric value specifying the point shape of censors. Default value is "+" (3), a sensible choice is "l" (124).
<code>censor.size</code>	numeric value specifying the point size of censors. Default is 4.5.
<code>pval</code>	logical value, a numeric or a string. If logical and TRUE, the p-value is added on the plot. If numeric, then the computed 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.
<code>pval.size</code>	numeric value specifying the p-value text size. Default is 5.
<code>pval.coord</code>	numeric vector, of length 2, specifying the x and y coordinates of the p-value. Default values are NULL.
<code>pval.parse</code>	logical. If TRUE, a custom p-value string supplied via <code>pval</code> is treated as a plotmath expression and parsed, so you can show italic/superscript p-values, e.g. <code>pval = "italic(P)==1.4~x~10^-6"</code> . The string must be a valid plotmath expression; the auto-computed p-value (e.g. <code>"p = 0.001"</code>) is plain text, not plotmath, so supply your own expression when using this. The test-name label from <code>pval.method</code> is never parsed. Default is FALSE (drawn literally).
<code>title</code>	main title
<code>xlab</code>	x axis label
<code>ylab</code>	y axis label
<code>xlim</code>	x axis limits e.g. <code>xlim = c(0, 1000)</code> .

<code>ylim</code>	y axis limits e.g. <code>xlim = c(0, 1)</code> .
<code>axes.offset</code>	logical value. Default is TRUE. If FALSE, set the plot axes to start at the origin.
<code>legend</code>	character specifying legend position. Allowed values are one of <code>c("top", "bottom", "left", "right", "none")</code> . Default is "top" side position. to remove the legend use <code>legend = "none"</code> . Legend position can be also specified using a numeric vector <code>c(x, y)</code> ; see details section.
<code>legend.title</code>	legend title.
<code>legend.labs</code>	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.
<code>risk.table</code>	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.
<code>risk.table.title</code>	The title to be used for the risk table.
<code>risk.table.pos</code>	character vector specifying the risk table position. Allowed options are one of <code>c("out", "in")</code> indicating 'outside' or 'inside' the main plot, respectively. Default value is "out".
<code>risk.table.col</code>	same as <code>tables.col</code> but for risk table only.
<code>risk.table.fontsize, fontsize</code>	font size to be used for the risk table and the cumulative events table.
<code>risk.table.y.text</code>	logical. Default is TRUE. If FALSE, risk table y axis tick labels will be hidden.
<code>risk.table.y.text.col</code>	logical. Default value is FALSE. If TRUE, risk table tick labels will be colored by strata.
<code>tables.height</code>	numeric value (in [0 - 1]) specifying the general height of all tables under the main survival plot.
<code>tables.y.text</code>	logical. Default is TRUE. If FALSE, the y axis tick labels of tables will be hidden.
<code>tables.y.text.col</code>	logical. Default value is FALSE. If TRUE, tables tick labels will be colored by strata.
<code>tables.col</code>	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 <code>tables.col = "strata"</code> .
<code>tables.theme</code>	function, ggplot2 theme name. Default value is <code>theme_survminer</code> . Allowed values include ggplot2 official themes: see theme . Note that, <code>tables.theme</code> is incremental to <code>ggtheme</code> .

<code>risk.table.height</code>	the height of the risk table on the grid. Increase the value when you have many strata. Default is 0.25. Ignored when <code>risk.table = FALSE</code> .
<code>surv.plot.height</code>	the height of the survival plot on the grid. Default is 0.75. Ignored when <code>risk.table = FALSE</code> .
<code>ncensor.plot</code>	logical value. If TRUE, the number of censored subjects at time t is plotted. Default is FALSE. Ignored when <code>cumcensor = TRUE</code> .
<code>ncensor.plot.title</code>	The title to be used for the censor plot. Used when <code>ncensor.plot = TRUE</code> .
<code>ncensor.plot.height</code>	The height of the censor plot. Used when <code>ncensor.plot = TRUE</code> .
<code>cumevents</code>	logical value specifying whether to show or not the table of the cumulative number of events. Default is FALSE. Can also be a character string ("absolute", "percentage" or "abs_pct"); "percentage" shows the count as a percent of the stratum size.
<code>cumevents.title</code>	The title to be used for the cumulative events table.
<code>cumevents.col</code>	same as <code>tables.col</code> but for the cumulative events table only.
<code>cumevents.y.text</code>	logical. Default is TRUE. If FALSE, the y axis tick labels of the cumulative events table will be hidden.
<code>cumevents.y.text.col</code>	logical. Default value is FALSE. If TRUE, the y tick labels of the cumulative events will be colored by strata.
<code>cumevents.height</code>	the height of the cumulative events table on the grid. Default is 0.25. Ignored when <code>cumevents = FALSE</code> .
<code>cumcensor</code>	logical value specifying whether to show or not the table of the cumulative number of censoring. Default is FALSE. Can also be a character string ("absolute", "percentage" or "abs_pct").
<code>cumcensor.title</code>	The title to be used for the cumcensor table.
<code>cumcensor.col</code>	same as <code>tables.col</code> but for cumcensor table only.
<code>cumcensor.y.text</code>	logical. Default is TRUE. If FALSE, the y axis tick labels of the cumcensor table will be hidden.
<code>cumcensor.y.text.col</code>	logical. Default value is FALSE. If TRUE, the y tick labels of the cumcensor will be colored by strata.
<code>cumcensor.height</code>	the height of the cumcensor table on the grid. Default is 0.25. Ignored when <code>cumcensor = FALSE</code> .
<code>surv.median.line</code>	character vector for drawing a horizontal/vertical line at median survival. Allowed values include one of <code>c("none", "hv", "h", "v")</code> . 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" so that weights 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). For an arbitrary Fleming-Harrington weight, use log.rank.weights = "FH" together with rho and gamma (weight $S(t^-)^\rho(1 - S(t^-))^\gamma$); e.g. rho = 1, gamma = 0 for early emphasis or rho = 0, gamma = 1 for late emphasis.
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 = NULL,
  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

<code>fit</code>	a named list whose elements are either survfit objects or <code>surv_summary()</code> data frames (the latter are drawn directly, as in <code>ggsurvplot_df()</code>). Note that the number-at-risk / cumulative tables and median lines are only available for survfit elements (they cannot be recomputed from a summary data frame).
<code>data</code>	the data frame used to compute survival curves. Optional; not needed when <code>fit</code> contains only <code>surv_summary()</code> data frames.
<code>risk.table</code>	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.
<code>risk.table.pos</code>	character vector specifying the risk table position. Allowed options are one of <code>c("out", "in")</code> indicating 'outside' or 'inside' the main plot, respectively. Default value is "out".
<code>cumevents</code>	logical value specifying whether to show or not the table of the cumulative number of events. Default is FALSE. Can also be a character string ("absolute", "percentage" or "abs_pct"); "percentage" shows the count as a percent of the stratum size.
<code>cumcensor</code>	logical value specifying whether to show or not the table of the cumulative number of censoring. Default is FALSE. Can also be a character string ("absolute", "percentage" or "abs_pct").
<code>tables.col</code>	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 <code>tables.col = "strata"</code> .
<code>tables.y.text</code>	logical. Default is TRUE. If FALSE, the y axis tick labels of tables will be hidden.
<code>tables.y.text.col</code>	logical. Default value is FALSE. If TRUE, tables tick labels will be colored by strata.
<code>ggtheme</code>	function, ggplot2 theme name. Default value is <code>theme_survminer</code> . Allowed values include ggplot2 official themes: see theme .
<code>tables.theme</code>	function, ggplot2 theme name. Default value is <code>theme_survminer</code> . Allowed values include ggplot2 official themes: see theme . Note that, <code>tables.theme</code> is incremental to <code>ggtheme</code> .
<code>keep.data</code>	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.
<code>risk.table.y.text</code>	logical. Default is TRUE. If FALSE, risk table y axis tick labels will be hidden.
<code>...</code>	other arguments to pass to the <code>ggsurvplot()</code> 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,
linejoin = "round",
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

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

	If <code>n.strata > 1</code>, the expected value is the grouping variable name. By default, survival curves are colored by strata using the argument <code>color = "strata"</code>, 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 <code>palette</code>.
<code>palette</code>	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. <code>c("blue", "red")</code> ; 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 .
<code>linetype</code>	line types. Allowed values includes i) "strata" for changing linetypes by strata (i.e. groups); ii) a numeric vector (e.g., <code>c(1, 2)</code>) or a character vector <code>c("solid", "dashed")</code> .
<code>break.x.by</code>	alias of <code>break.time.by</code> . Numeric value controlling x axis breaks. Default value is NULL.
<code>break.time.by</code>	numeric value controlling time axis breaks. Default value is NULL.
<code>break.y.by</code>	same as <code>break.x.by</code> but for y axis.
<code>surv.scale</code>	scale transformation of survival curves. Allowed values are "default" or "percent".
<code>surv.geom</code>	survival curve style. Is the survival curve entered a step function (geom_step) or a smooth function (geom_line).
<code>linejoin</code>	line join style for the survival curve, passed to the survival geom. Default is "round" (unchanged). Use "mitre" for sharp, precisely-marked corners at event times (requires a ggplot2 version that passes <code>linejoin</code> through geom_step).
<code>xscale</code>	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 <code>c("d_m", "d_y", "m_d", "m_y", "y_d", "y_m")</code>, where d = days, m = months and y = years. For example, <code>xscale = "d_m"</code> will transform labels from days to months; <code>xscale = "m_y"</code>, will transform labels from months to years.
<code>conf.int</code>	logical value. If TRUE, plots confidence interval.
<code>conf.int.fill</code>	fill color to be used for confidence interval.
<code>conf.int.style</code>	confidence interval style. Allowed values include <code>c("ribbon", "step")</code> .
<code>conf.int.alpha</code>	numeric value specifying fill color transparency. Value should be in [0, 1], where 0 is full transparency and 1 is no transparency.
<code>censor</code>	logical value. If TRUE, censors will be drawn.
<code>censor.shape</code>	character or numeric value specifying the point shape of censors. Default value is "+" (3), a sensible choice is "I" (124).
<code>censor.size</code>	numeric value specifying the point size of censors. Default is 4.5.
<code>title</code>	main title

xlab	x axis label
ylab	y axis label
xlim	x axis limits e.g. <code>xlim = c(0, 1000)</code> .
ylim	y axis limits e.g. <code>ylim = c(0, 1)</code> .
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 <code>c("top", "bottom", "left", "right", "none")</code> . Default is "top" side position. to remove the legend use <code>legend = "none"</code> . Legend position can be also specified using a numeric vector <code>c(x, y)</code> ; 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 <code>theme_survminer</code> . Allowed values include ggplot2 official themes: see theme .
...	other arguments to be passed i) to ggplot2 <code>geom_*()</code> functions such as <code>linetype</code> , <code>size</code> , ii) or to the function <code>ggpar()</code> 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.size = 5,
  pval.coord = NULL,
  pval.method.coord = NULL,
  pval.in.label = FALSE,
  pval.stratified = FALSE,
  p.adjust.method = "none",
  p.adjust.label = "adj.p =",
  nrow = NULL,
  ncol = NULL,
  scales = "fixed",
  short.panel.labs = FALSE,
  panel.labs = NULL,
  labeller = 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,
  risk.table = FALSE,

```

```

    risk.table.height = 0.25,
    surv.plot.height = 0.75,
    tables.y.text = TRUE,
    tables.col = "black",
    tables.theme = theme_survminer(),
    fontsize = 4.5,
    ...
)

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

```

Arguments

<code>fit</code>	an object of class <code>survfit</code> .
<code>data</code>	a dataset used to fit survival curves. If not supplied then data will be extracted from 'fit' object.
<code>facet.by</code>	character vector, of length 1 or 2, specifying grouping variables for faceting the plot. Should be in the data.
<code>color</code>	color to be used for the survival curves. • If the number of strata/group (<code>n.strata</code>) = 1, the expected value is the color name. For example <code>color = "blue"</code>. • If <code>n.strata > 1</code>, the expected value is the grouping variable name. By default, survival curves are colored by strata using the argument <code>color = "strata"</code>, 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 <code>palette</code>.
<code>palette</code>	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. <code>c("blue", "red")</code> ; and scientific journal palettes from ggsci R package, e.g.: "npg", "aaas", "lancet", "jco", "ucscgb", "uchicago", "simpsons" and "rickandmarty". See details section for more information. Can be also a numeric vector of length(<code>groups</code>); in this case a basic color palette is created using the function palette .
<code>legend.labs</code>	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.
<code>pval</code>	logical value. If TRUE, a log-rank p-value is computed and displayed for each panel. Note that, unlike ggsurvplot() , a numeric or character pval is not substituted here (there is one p-value per panel); such a value is ignored with a warning.
<code>pval.method</code>	whether to add a text with the test name used for calculating the pvalue, that corresponds to survival curves' comparison - used only when <code>pval=TRUE</code>
<code>pval.size</code>	numeric value specifying the p-value text size. Default is 5.
<code>pval.coord</code>	numeric vector, of length 2, specifying the x and y coordinates of the p-value. Default values are NULL.

<code>pval.method.coord</code>	the same as <code>pval.coord</code> but for displaying <code>log.rank.weights</code> name
<code>pval.in.label</code>	logical value. Default is FALSE. If TRUE, the per-panel p-value is appended to the panel strip label instead of being drawn inside the panel. Requires <code>pval = TRUE</code> and a single <code>facet.by</code> variable (with two faceting variables the p-value is drawn on the panels instead, with a warning). If a <code>labeller</code> is also supplied it takes precedence and the p-value is not added to the label.
<code>pval.stratified</code>	logical value. Default is FALSE. If TRUE, a single pooled stratified log-rank p-value – <code>survdifff(Surv(time, status) ~ group + strata(facet.by))</code> , the group effect adjusting for the faceting variable – is shown as a figure-level subtitle. This is the statistic the whole figure illustrates: the per-panel p-values (<code>pval = TRUE</code>) are within-facet and several may be non-significant even when the pooled effect is strong. <code>pval.stratified</code> is independent of <code>pval</code> and can be combined with it. Several <code>facet.by</code> variables are combined into one <code>strata()</code> . The pooled test is the log-rank / Peto family (a weighted Fleming-Harrington method is ignored, as <code>survdifff</code> has no stratified Fleming-Harrington). Caveat: the stratified log-rank assumes the group effect is consistent across the facet strata; it sums the score over strata, so opposite-direction effects cancel and it is <i>not</i> a test of a group-by-facet interaction (Klein & Moeschberger, 2003).
<code>p.adjust.method</code>	method for adjusting the per-panel p-values for multiple comparisons across panels, passed to <code>p.adjust()</code> (e.g. "BH", "bonferroni", "holm"). The default "none" shows the raw per-panel p-values. When an adjustment method is used the displayed text is prefixed with <code>p.adjust.label</code> .
<code>p.adjust.label</code>	the label placed before the adjusted p-value when <code>p.adjust.method</code> is not "none". The default is "adj.p =" (e.g. "adj.p = 0.03"); set it to customise the wording, e.g. "p.adj =" or "q =". For a very small p-value the label's trailing "=" is replaced by "<" ("adj.p < 0.0001").
<code>nrow, ncol</code>	Number of rows and columns in the panel. Used only when the data is faceted by one grouping variable.
<code>scales</code>	should axis scales of panels be fixed ("fixed", the default), free ("free"), or free in one dimension ("free_x", "free_y").
<code>short.panel.labs</code>	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.
<code>panel.labs</code>	a list of one or two character vectors to modify facet label text. For example, <code>panel.labs = list(sex = c("Male", "Female"))</code> specifies the labels for the "sex" variable. For two grouping variables, you can use for example <code>panel.labs = list(sex = c("Male", "Female"), rx = c("Obs", "Lev", "Lev2"))</code> .
<code>labeller</code>	a labeller function/specification, as in <code>facet_wrap</code> (e.g. <code>ggplot2::label_both</code> or <code>ggplot2::as_labeller(c(...))</code>), controlling how the panel strip labels are formatted. Default NULL keeps the current labels. When supplied it takes precedence over <code>short.panel.labs</code> ; it composes with <code>panel.labs</code> (which renames the underlying data levels, so the labeller sees the renamed values).

<code>panel.labs.background</code>	a list to customize the background of panel labels. Should contain the combination of the following elements: • <code>color</code>, <code>linetype</code>, <code>size</code>: background line color, type and size • <code>fill</code>: background fill color. For example, <code>panel.labs.background = list(color = "blue", fill = "pink")</code>.
<code>panel.labs.font</code>	a list of aesthetics 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.
<code>panel.labs.font.x</code> , <code>panel.labs.font.y</code>	same as <code>panel.labs.font</code> but for x and y direction, respectively.
<code>risk.table</code>	logical value. Default is FALSE. If TRUE, a number-at-risk table is drawn below the faceted curves, faceted with the same structure so each panel's table aligns under its curves. Supported for a single <code>facet.by</code> variable; with two it warns and returns the plot without a table (a two-way <code>facet_grid</code> shares one y axis per row, so the per-panel strata cannot be labelled correctly). When <code>risk.table</code> is not FALSE the function returns an aligned gtable (printed with <code>print()</code>); the default (FALSE) returns the faceted ggplot unchanged. In the faceted table the strata rows are identified by their text labels in a single colour (not by the curve colours), because a panel that drops a stratum would otherwise recolour the remaining rows by position and mislabel them. The table rows are labelled by the grouping variable's own values; <code>legend.labs</code> relabels the curve legend but not the table rows.
<code>risk.table.height</code> , <code>surv.plot.height</code>	the relative heights of the risk table and the survival plot when <code>risk.table</code> is drawn. Defaults are 0.25 and 0.75.
<code>tables.y.text</code>	logical value. Default is TRUE. In a faceted risk table the rows are identified by their text labels (the curve colours are not a reliable identifier across facets), so FALSE is not supported here: the labels are kept and a warning is issued.
<code>tables.col</code> , <code>tables.theme</code> , <code>fontsize</code>	colour, theme and font size of the risk table, as in ggsurvplot.
<code>...</code>	other arguments to pass to the function ggsurvplot.
<code>x</code>	an object of class <code>ggsurvplot_facet</code> (the gtable returned by <code>ggsurvplot_facet()</code> when <code>risk.table</code> is drawn).
<code>newpage</code>	logical. If TRUE (default), a new page is drawn before the plot.

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)

# P-value in the panel labels (single faceting variable)
#::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
ggsurvplot_facet(fit, colon, facet.by = "rx",
                 palette = "jco", pval = TRUE, pval.in.label = TRUE)

# Pooled stratified log-rank p (figure-level), with the per-panel p-values
#::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
ggsurvplot_facet(fit, colon, facet.by = "rx",
                 palette = "jco", pval = TRUE, pval.stratified = TRUE)

# Faceted number-at-risk table
#::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
ggsurvplot_facet(fit, colon, facet.by = "rx",
                 palette = "jco", risk.table = 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

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

Details

`ggsurvplot_group_by()` works as follow:

1. Create a grouped data sets using the function `surv_group_by()`, $\rightarrow$ list of data sets
2. Map `surv_fit()` to each nested data $\rightarrow$ Returns a list of survfit objects
3. Map `ggsurvplot()` to each survfit object $\rightarrow$ 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	<i>Plot a List of Survfit Objects</i>
-----------------	---------------------------------------

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

<code>fit</code>	a list of survfit objects.
<code>data</code>	data used to fit survival curves. Can be also a list of same length than <code>fit</code> .
<code>title</code>	title of the plot. Can be a character vector or a list of titles of same length than <code>fit</code> . If <code>title</code> is not specified and <code>fit</code> is a named list, then the names of <code>fit</code> list are used as title.
<code>legend.labs</code>	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 <code>fit</code> is a list.
<code>legend.title</code>	legend title for each plot. Can be a character vector or a list of titles of same length than <code>fit</code> .
<code>...</code>	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	<i>Multiple Comparisons of Survival Curves</i>
-------------------	--

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,
  method = "survdiff",
  ref.group = NULL,
  detailed = FALSE
)
```

Arguments

formula	a formula expression as for other survival models, of the form <code>Surv(time, status) ~ predictors</code> .
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 <code>p.adjust.method = "none"</code> .
na.action	a missing-data filter function. Default is <code>options()\$na.action</code> .
rho	a scalar parameter that controls the type of test (used when <code>method = "survdiff"</code>). Allowed values include 0 (for Log-Rank test) and 1 (for peto & peto test).
method	the log-rank test to use, matching the method argument of surv_pvalue . Default "survdiff" uses <code>survival::survdiff()</code> (controlled by rho). A weighted log-rank test can be requested by its weight code, full name or abbreviation: "n"/Gehan-Breslow, "sqrtn"/Tarone-Ware, "S1"/Peto-Peto, "S2"/modified Peto-Peto, or "FH_p=1_q=1"/Fleming-Harrington. Weighted methods do not support <code>strata()</code> terms.
ref.group	optional character string naming a single grouping level to use as the common reference/control. When supplied, every other group is compared against this one group only (rather than all pairwise comparisons), and the p-values are adjusted over just those comparisons – useful for a many-treatments-vs-one-control design, which needs a smaller multiple-testing correction. The default NULL performs all pairwise comparisons.

detailed logical. If TRUE, attach a per-pair table `res$detailed` with the test statistic (chi-square) and its degrees of freedom, the raw and adjusted p-values, and a Cox proportional-hazards hazard ratio with 95% confidence interval for each pair (the hazard of group2 relative to group1). The default FALSE returns the usual p-value matrix only. The hazard ratio and its interval can be unstable or infinite for a pair with (near-)complete separation; such a pair returns NA if the Cox model does not fit.

Value

Returns an object of class "pairwise.htest", which is a list containing the p values. With `detailed = TRUE` it also carries a detailed data frame of per-pair statistics, degrees of freedom, p-values and Cox hazard ratios.

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_adtte

Analysis-Ready Survival Data from a CDISC ADTTE Dataset

Description

Prepares a CDISC ADaM time-to-event dataset (ADTTE) for survival analysis, guarding against the classic censoring-flip error. In ADTTE the censoring flag CNSR is coded **0 = event**, ≥ 1 = **censored** (values 1-4 give the censoring reason) – the opposite of `Surv()`, which wants 1 = event. Passing CNSR straight into `Surv()` silently inverts every result. `surv_adtte()` derives the correct event

indicator (event = as.integer(CNSR == 0)), filters to one analysis parameter, validates the input, and returns the data with an event column ready for `surv_fit()` / `ggsurvplot()`.

Note the derivation `CNSR == 0` (not `1 - CNSR`): the latter is only correct for binary CNSR and breaks for the standard multi-valued censoring-reason codes.

Usage

```
surv_adtte(
  data,
  paramcd = NULL,
  cnsr = "CNSR",
  aval = "AVAL",
  param = "PARAMCD",
  subject = "USUBJID",
  event.name = "event",
  event.ref = NULL,
  overwrite = FALSE
)
```

Arguments

<code>data</code>	an ADTTE data frame.
<code>paramcd</code>	the analysis parameter to keep, matched against the <code>param</code> column (e.g. "OS", "PFS"). If <code>NULL</code> and several parameters are present, an error is raised (a time-to-event analysis is per parameter).
<code>cnsr, aval, param</code>	column names of the censoring flag, analysis value (time) and parameter code. Defaults are the ADaM standard "CNSR", "AVAL", "PARAMCD".
<code>subject</code>	column name of the subject identifier, used only to warn about duplicate records (ADTTE has one record per subject per parameter). Default "USUBJID"; ignored if absent.
<code>event.name</code>	name of the 0/1 event column to add. Default "event".
<code>event.ref</code>	optional name of an existing event-indicator column (<code>1 = event</code>) to cross-check the derivation against; a warning is issued when the derived event disagrees with it (catches an already-flipped or nonstandard coding). There is no standard ADTTE event flag, so this is opt-in.
<code>overwrite</code>	logical. If <code>event.name</code> already exists in <code>data</code> , overwrite it (<code>TRUE</code>) or error (<code>FALSE</code> , default).

Value

`data`, filtered to `paramcd` and with the event column added (rows in input order, row names reset). The event and censoring counts are attached as `attr(x, "adtte")` and reported in a message.

See Also

[surv_fit](#), [ggsurvplot](#). The `ggsurvfit::Surv_CNSR()` and `visR::Surv_CNSR()` functions build a `Surv` object from `CNSR`; `surv_adtte()` instead returns a tidy data frame (filtered and validated), and derives the event with `CNSR == 0` rather than `1 - CNSR`.

Examples

```
# example ADTTE bundled with survminer (CNSR: 0 = event, >= 1 = censored)
data("adtte")
head(adtte)

# derive the event, filter to overall survival, and plot
os <- surv_adtte(adtte, paramcd = "OS")
fit <- surv_fit(survival::Surv(AVAL, event) ~ TRT01P, data = os)
ggsurvplot(fit, data = os, pval = TRUE, risk.table = TRUE)
```

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

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

<code>formula</code>	survival formula. See survfit.formula . Can be a list of formula. Named lists are recommended.
<code>data</code>	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() .
<code>group.by</code>	a grouping variables to group the data set by. A character vector containing the name of grouping variables. Should be of length ≤ 2 .
<code>match.fd</code>	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 <code>match.fd = TRUE</code> .
<code>...</code>	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

<code>data</code>	a data frame
<code>grouping.vars</code>	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 column named "data", which is a named list of data subsets created by the grouping variables. The list names are created by concatenating 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. The confidence limits are taken at the confidence level the survfit was built with (95% by default); a fit stored without confidence limits (e.g. `conf.type = "none"`) returns NA limits.

Usage

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

Arguments

<code>fit</code>	A survfit object. Can be also a list of survfit objects.
<code>combine</code>	logical value. Used only when <code>fit</code> 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: lower confidence limit of the median
- upper: upper confidence limit of the median

The confidence limits use the fit's own confidence level (95% by default), and are NA for a fit built without confidence limits. 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)
```

Description

Estimates the median follow-up time using the reverse Kaplan-Meier method (Schemper & Smith, 1996), also called the "potential" follow-up time. The roles of the event and censoring indicators are swapped – censored subjects are treated as the events – and the median of the resulting Kaplan-Meier curve is reported, per group.

This is the follow-up counterpart of `surv_median()`: the latter returns the median *survival* time, this returns the median *follow-up* time. Reporting the median follow-up is standard practice in clinical publications because the naive median of the observed times under-estimates follow-up (deaths stop a subject's follow-up early).

Usage

```
surv_median_followup(fit, data = NULL, conf.level = NULL)
```

Arguments

<code>fit</code>	a <code>survfit</code> object (<code>survival::survfit()</code> or <code>survminer</code> 's <code>surv_fit()</code>) built from right-censored <code>Surv(time, status)</code> data.
<code>data</code>	the data frame used to fit the survival curves. If not supplied it is extracted from <code>fit</code> .
<code>conf.level</code>	the confidence level for the limits, a single number in (0, 1). Default <code>NULL</code> uses the confidence level the <code>fit</code> was built with (95% by default). The <code>fit</code> 's confidence-interval type (<code>conf.type</code>) is honoured as well. Named <code>conf.level</code> rather than <code>conf.int</code> so that, as everywhere else in <code>survminer</code> , <code>conf.int</code> means "draw the band" and <code>conf.level</code> means "at what level".

Details

The estimate is computed entirely from base `survival`: the curve is refit on the reversed event indicator (each subject's event/censor status flipped) with the same grouping as `fit`, and the median (with confidence limits) is read from its summary. The status is read from the `fit`'s `Surv` response, so any right-censored coding works (0/1, 1/2, or logical). The confidence limits are the standard Kaplan-Meier limits at the `fit`'s own confidence level and interval type; they are `NA` when the reverse curve does not drop below the relevant bound (e.g. a small group whose upper limit is not reached, or a `fit` built with `conf.type = "none"`). Only right-censored `Surv(time, status)` data are supported.

Value

A data frame with one row per group and the columns:

- `strata`: group name ("All" for an ungrouped `fit`)
- `median`: median follow-up time
- `lower`: lower confidence limit
- `upper`: upper confidence limit

References

Schemper M, Smith TL (1996). A note on quantifying follow-up in studies of failure time. *Controlled Clinical Trials* 17(4):343-346.

See Also

[surv_median\(\)](#)

Examples

```
library(survival)

# Grouped fit: median follow-up per sex
fit <- surv_fit(Surv(time, status) ~ sex, data = lung)
surv_median_followup(fit)

# Contrast with the median survival time (a different quantity)
surv_median(fit)

# Ungrouped fit: overall median follow-up
fit.null <- surv_fit(Surv(time, status) ~ 1, data = lung)
surv_median_followup(fit.null)
```

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 = "survdif",
  test.for.trend = FALSE,
  combine = FALSE,
  ...
)
```

Arguments

<code>fit</code>	A survfit object. Can be also a list of survfit objects.
<code>data</code>	data frame used to fit survival curves. Can be also a list of data.

method	method to compute survival curves. Default is "survdif" (or "log-rank"). Allowed values are one of: • "survdif", 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 acronym LR, GB, ... Case insensitive partial match is allowed. For an arbitrary Fleming-Harrington $G(\rho, \gamma)$ weight, use method = "FH" together with rho and gamma (passed via . . .): the weight is $S(t^-)^\rho(1 - S(t^-))^\gamma$. FH(0, 0) is the log-rank test, FH(1, 0) emphasises early differences, FH(0, 1) late differences and FH(1, 1) the middle. See weighted_logrank() for a batch version. Note: for backward compatibility a bare method = "FH" (with no rho/gamma) is FH(1, 1); pass rho = 0, gamma = 0 for the log-rank test. 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. One useful argument is pval.digits: the number of significant digits used to format the p-value in the returned pval.txt. Default is 2 (e.g. "p = 0.0013"); set pval.digits = 3 for "p = 0.00131", etc. P-values below 0.0001 are still reported as "p < 0.0001". The remaining arguments (pval, pval.coord, pval.method.coord) are only used internally to specify custom pvalue and pvalue-method coordinates on the survival plot; normally users don't need them.

Value

Return a data frame with the columns (variable, pval, method, pval.txt, statistic and df). 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
- statistic: the test statistic (chi-square)
- df: its degrees of freedom
- 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 annotation 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	<i>Theme for Survminer Plots</i>
-----------------	----------------------------------

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 = "", ...)
```

```
theme_surv_classic(
  base_size = 12,
  base_family = "",
  font.main = c(16, "bold", "black"),
  font.submain = c(15, "plain", "black"),
  font.x = c(14, "bold", "black"),
  font.y = c(14, "bold", "black"),
  font.caption = c(15, "plain", "black"),
  font.tickslab = c(12, "plain", "black"),
  legend = c("bottom", "top", "left", "right", "none"),
  font.legend = c(11, "plain", "black"),
  ...
)
```

```
theme_surv_minimal(
  base_size = 12,
  base_family = "",
  font.main = c(15, "plain", "black"),
  font.submain = c(14, "plain", "black"),
  font.x = c(13, "plain", "black"),
  font.y = c(13, "plain", "black"),
  font.caption = c(13, "plain", "black"),
```

```

font.tickslab = c(11, "plain", "black"),
legend = c("bottom", "top", "left", "right", "none"),
font.legend = c(10, "plain", "black"),
...
)

theme_surv_bold(
  base_size = 14,
  base_family = "",
  font.main = c(20, "bold", "black"),
  font.submain = c(17, "bold", "black"),
  font.x = c(16, "bold", "black"),
  font.y = c(16, "bold", "black"),
  font.caption = c(15, "plain", "black"),
  font.tickslab = c(14, "bold", "black"),
  legend = c("top", "bottom", "left", "right", "none"),
  font.legend = c(14, "plain", "black"),
  ...
)

```

Arguments

<code>base_size</code>	base font size
<code>base_family</code>	base font family
<code>font.main</code> , <code>font.submain</code> , <code>font.caption</code> , <code>font.x</code> , <code>font.y</code> , <code>font.tickslab</code> , <code>font.legend</code>	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 <code>font.x = c(14, "bold", "red")</code> . Use <code>font.x = 14</code> , to change only font size; or use <code>font.x = "bold"</code> , to change only font face.
<code>legend</code>	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 <code>legend = "none"</code> . Legend position can be also specified using a numeric vector c(x, y); see details section.
<code>...</code>	additional arguments passed to the function <code>theme_survminer()</code> .

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.
- `theme_surv_classic()`: a clean, publication-classic look for survival curves: [theme_survminer\(\)](#) with bold axis titles, no panel gridlines, crisp axis lines and a bottom legend. A good default for a manuscript figure.

- `theme_surv_minimal()`: a minimal, airy look: no axis lines or ticks and only a faint horizontal gridline, for a clean single-panel figure. Pairs well with an inside legend.
- `theme_surv_bold()`: a bold, slide-friendly look: large fonts and heavy axis lines so the figure stays legible when projected on a poster or in a talk.

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

- * **ADTTE**
 - surv_adtte, 79
- * **ADaM**
 - surv_adtte, 79
- * **CDISC**
 - surv_adtte, 79
- * **CNSR**
 - surv_adtte, 79
- * **censoring**
 - surv_adtte, 79
- * **survival**
 - surv_adtte, 79
- + .ggsurv (add_ggsurvplot), 3
- %++% (add_ggsurvplot), 3
- add_ggsurvplot, 3
- adtte, 4
- arrange_ggsurvplots, 5, 75
- arrangeGrob, 9, 22, 27
- BMT, 7
- BRCAOV.survInfo, 8
- build_ggsurvplot, 9
- cox.zph, 24, 26
- coxph, 12, 19–22, 24, 26, 32
- coxph.object, 12, 20, 22
- cuminc, 16, 17
- customize_labels, 10
- element_markdown, 10
- facet_wrap, 72
- flexsurvreg, 48
- geom_hline, 20
- geom_line, 68
- geom_smooth, 20
- geom_step, 68
- get_palette, 30
- ggadjustedcurves, 11
- ggcompetingrisks, 16
- ggcoxdiagnostics, 19, 25
- ggcoxfunctional, 21
- ggcoxnp, 23
- ggcoxzph, 25, 26
- ggcumcensor (ggrisktable), 40
- ggcumevents (ggrisktable), 40
- ggflexsurvplot, 28, 49
- ggforest, 29, 32, 33, 35
- ggforest_models, 32
- ggforest_subgroup, 32, 33, 33
- gglandmark, 36, 40, 45
- ggmilestone, 37, 38, 45
- ggpar, 10, 14, 18, 20, 22, 25, 27, 41, 47, 53, 56, 64, 69
- ggplotGrob, 9
- ggrisktable, 40
- ggrmst, 25, 37, 40
- ggrmst (ggrmst_difference), 44
- ggrmst_difference, 39, 44
- ggsave, 6, 9
- ggsurvevents, 46
- ggsurvparametric, 29, 48
- ggsurvplot, 3, 9, 10, 13–15, 17, 29, 36–40, 45, 48, 49, 49, 59, 64–66, 71, 73–76, 80, 81, 84, 93, 97
- ggsurvplot_add_all, 49, 52, 58
- ggsurvplot_arguments, 53, 60
- ggsurvplot_combine, 49, 52, 64
- ggsurvplot_df, 50, 65, 66
- ggsurvplot_facet, 49, 52, 70
- ggsurvplot_group_by, 49, 52, 74
- ggsurvplot_list, 49, 50, 76
- ggsurvtable, 41
- ggsurvtable (ggrisktable), 40
- grid.arrange, 22, 27, 53, 82
- grid.draw, 9, 53
- grid.draw.ggcoxnp (ggcoxnp), 23
- grid.draw.ggsurvplot (ggsurvplot), 49

grob, 22, 27
 lowess, 21, 22
 marrangeGrob, 6
 myeloma, 77

 p.adjust, 72, 78
 pairwise_survdif, 78
 palette, 13, 42, 51, 61, 68, 71
 plot.cox.zph, 26
 plot.surv_cutpoint (surv_cutpoint), 81
 plotmath, 54
 print.ggcoxfunctional
 (ggcoxfunctional), 21
 print.ggcoxnp (ggcoxnp), 23
 print.ggcoxzph (ggcoxzph), 26
 print.ggsurvplot (ggsurvplot), 49
 print.ggsurvplot_facet
 (ggsurvplot_facet), 70
 print.plot_surv_cutpoint
 (surv_cutpoint), 81
 print.surv_cutpoint (surv_cutpoint), 81

 residuals.coxph, 20

 summary.surv_cutpoint (surv_cutpoint),
 81
 Surv, 5, 22, 47, 79
 surv_adjustedcurves (ggadjustedcurves),
 11
 surv_adtte, 4, 5, 79
 surv_categorize (surv_cutpoint), 81
 surv_cutpoint, 81
 surv_fit, 36–38, 45, 49, 75, 80, 81, 83, 89
 surv_group_by, 75, 83, 84, 86
 surv_median, 39, 46, 84, 87, 89, 90
 surv_median_followup, 88
 surv_pvalue, 78, 84, 90, 97
 surv_summary, 39, 46, 49, 65, 66, 92
 survfit, 17, 47, 83, 92
 survfit.formula, 49, 66, 84
 survreg, 48

 theme, 13, 17, 20, 22, 27, 43, 52, 62, 64, 65,
 69, 82
 theme_classic, 82
 theme_cleantable (theme_survminer), 94
 theme_surv_bold (theme_survminer), 94
 theme_surv_classic, 53

 theme_surv_classic (theme_survminer), 94
 theme_surv_minimal (theme_survminer), 94
 theme_survminer, 3, 17, 25, 33, 35, 43, 48,
 52, 62, 64, 65, 69, 94, 95
 tibble, 87

 weighted_logrank, 91, 96