Package: weightedsurv (via r-universe)

October 29, 2026

Title Survival Analysis with Subject-Specific (Case Weights) and Time-Dependent Weighting

Version 0.1.0

Description Provides survival analysis functions with support for time-dependent and subject-specific (e.g., propensity score) weighting. Implements weighted estimation for Cox models, Kaplan-Meier survival curves, and treatment differences with point-wise and simultaneous confidence bands. Includes restricted mean survival time (RMST) comparisons evaluated across all potential truncation times with both point-wise and simultaneous confidence bands. See Cole, S. R. & Hernán, M. A. (2004) <doi:10.1016/j.cmpb.2003.10.004> for methodological background.

License MIT + file LICENSE

URL https://larry-leon.github.io/weightedsurv/, https://github.com/larry-leon/weightedsurv

BugReports https://github.com/larry-leon/weightedsurv/issues

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Imports ggplot2, graphics, rlang, stats, survival, utils

Suggests adjustedCurves, data.table, DiagrammeR, dplyr, gt, knitr, pkgdown, pammtools, rmarkdown

VignetteBuilder knitr

Repository https://larry-leon.r-universe.dev

Date/Publication 2026-02-12 18:28:14 UTC

RemoteUrl https://github.com/larry-leon/weightedsurv

RemoteRef HEAD

RemoteSha edd61a2cecf4b08352719af0a24b10fa1e0b041c

Contents

add_legends . . . 3
add_median_annotation . . . 4
add_risk_table . . . 5
calculate_risk_event_counts . . . 6
check_km_curve . . . 6
check_results . . . 7
ci_cox . . . 8
count_weighted . . . 9
cox_rhogamma . . . 9
cox_rhogamma_resample . . . 12
cox_score_rhogamma . . . 13
create_baseline_table . . . 14
cumulative_rmst_bands . . . 15
df_counting . . . 17
extract_and_calc_weights . . . 20
extract_group_data . . . 21
find_cox_root . . . 22
format_pval . . . 22
get_censoring_and_events . . . 23
get_dfcounting . . . 23
get_event_risk_matrices . . . 24
get_riskpoints . . . 24
get_validated_weights . . . 25
get_weights . . . 25
KM_diff . . . 26
KM_estimates . . . 28
KM_plot_2sample_weighted_counting . . . 29
km_quantile . . . 32
km_quantile_table . . . 33
kmq_calculations . . . 34
N_rhogamma . . . 34
plot_km . . . 35
plot_km_confint_polygon . . . 35
plot_km_curves_counting . . . 36
plot_weight_schemes . . . 38
plot_weighted_km . . . 39
plotKM.band_subgroups . . . 39
resampling_survival . . . 42
risk_weighted . . . 43
safe_run . . . 43
score_calculation . . . 44
validate_input . . . 44
validate_scheme_params . . . 45
wlr_cumulative . . . 45
wlr_dhat_estimates . . . 46
wt.rg.S . . . 47

Index **50**

`add_legends` *Add legends to KM plot*

Description

Adds legends for Cox model, log-rank test, and arms to a Kaplan-Meier plot.

Usage

```
add_legends(
  dfcount,
  show.cox,
  cox.cex,
  put.legend.cox,
  show.logrank,
  logrank.cex,
  put.legend.lr,
  show_arm_legend,
  arms,
  col.1,
  col.0,
  ltys,
  lwds,
  arm.cex,
  put.legend.arms
)
```

Arguments

`dfcount`	List with results, typically output from a survival analysis function. Should contain elements such as `cox_results`, `z.score`, and `zlogrank_text`.
`show.cox`	Logical; show Cox legend.
`cox.cex`	Numeric; Cox legend size.
`put.legend.cox`	Character; Cox legend position (e.g., "topright").
`show.logrank`	Logical; show logrank legend.
`logrank.cex`	Numeric; logrank legend size.
`put.legend.lr`	Character; logrank legend position (e.g., "topleft").
`show_arm_legend`	Logical; show arm legend.
`arms`	Character vector of arm labels.
`col.1`	Color for group 1.
`col.0`	Color for group 0.
`ltys`	Line types.

lwds	Line widths.
arm.cex	Numeric; arm legend size.
put.legend.arms	Character; arm legend position (e.g., "left").

Value

Invisibly returns NULL. Used for plotting side effects.

`add_median_annotation` *Add median annotation to KM plot*

Description

Adds median survival annotation to a Kaplan-Meier plot.

Usage

```
add_median_annotation(
  medians_df,
  med.digits,
  med.cex,
  med.font,
  xmed.fraction,
  ymed.offset
)
```

Arguments

medians_df	Data frame with quantile results. Should contain columns `quantile`, `lower`, `upper`, and `group`.
med.digits	Integer; number of digits to display for median and confidence interval.
med.cex	Numeric; text size for median annotation.
med.font	Integer; font for median annotation.
xmed.fraction	Numeric; fraction of the x-axis for annotation placement (e.g., 0.8 for 80\% to the right).
ymed.offset	Numeric; offset from the top of the plot for annotation placement.

Value

Invisibly returns NULL. Used for plotting side effects.

`add_risk_table` *Add risk table annotation to KM plot*

Description

Adds risk set counts for two groups to a Kaplan-Meier plot.

Usage

```
add_risk_table(
  risk.points,
  rpoints0,
  rpoints1,
  col.0,
  col.1,
  risk.cex,
  ymin,
  risk_offset,
  risk_delta,
  y.risk0,
  y.risk1
)
```

Arguments

`risk.points`	Numeric vector of risk time points.
`rpoints0`	Numeric vector of risk set counts for group 0.
`rpoints1`	Numeric vector of risk set counts for group 1.
`col.0`	Color for group 0.
`col.1`	Color for group 1.
`risk.cex`	Numeric; text size for risk table.
`ymin`	Numeric; minimum y value.
`risk_offset`	Numeric; offset for risk table.
`risk_delta`	Numeric; delta for risk table.
`y.risk0`	Numeric; y position for group 0 risk table.
`y.risk1`	Numeric; y position for group 1 risk table.

Value

Invisibly returns NULL. Used for plotting side effects.

calculate_risk_event_counts *Calculate risk set and event counts at time points*

Description

Calculates risk set and event counts for a group at specified time points, with variance estimation.

Usage

```
calculate_risk_event_counts(U, D, W, at_points, draws = 0, seedstart = 816951)
```

Arguments

U	Numeric vector of times for group.
D	Numeric vector of event indicators for group.
W	Numeric vector of weights for group.
at_points	Numeric vector of time points.
draws	Number of draws for variance estimation (default 0).
seedstart	Random seed for draws (default 816951).

Value

List with ybar (risk set counts), nbar (event counts), sig2w_multiplier (variance term).

check_km_curve *Check KM curve for validity*

Description

Checks that a Kaplan-Meier curve is valid (values in [0,1], non-increasing).

Usage

```
check_km_curve(S.KM, group_name = "Group", stop_on_error = TRUE)
```

Arguments

S.KM	Numeric vector of survival probabilities.
group_name	Character; name of the group.
stop_on_error	Logical; whether to stop on error (default TRUE).

Value

None. Stops or warns if invalid.

check_results *Check and Compare Statistical Test Results*

Description

Calculates and compares three equivalent test statistics from weighted survival analysis: the squared standardized weighted log-rank statistic, the log-rank chi-squared statistic, and the squared Cox model z-score. These should be approximately equal under correct implementation.

Usage

```
check_results(dfcount, verbose = TRUE)
```

Arguments

dfcount	A list or data frame from df_counting containing: **lr** Weighted log-rank statistic **sig2_lr** Variance of the weighted log-rank statistic **z.score** Standardized z-score from Cox model **logrank_results** List containing chisq, the log-rank chi-squared statistic
verbose	Logical; if TRUE, prints the comparison table to the console. Default: TRUE.

Details

This function serves as a diagnostic tool to verify computational consistency. The three statistics should be numerically equivalent (within rounding error):

$$(lr/\sqrt{sig2_lr})^2 \approx logrank_chisq \approx z.score^2$$

Discrepancies between these values may indicate:

- Numerical instability in variance estimation
- Incorrect weighting scheme application
- Data processing errors

Value

A data frame with one row and three columns, returned invisibly:

zlr_sq Squared standardized weighted log-rank: $(lr/\sqrt{sig2_lr})^2$

logrank_chisq Chi-squared statistic from log-rank test

zCox_sq Squared z-score from Cox model: $z.score^2$

Note

This function is primarily used for package development and validation. End users typically don't need to call it directly.

See Also

df_counting for generating the input object

Examples

```
# After running df_counting
library(survival)
data(veteran)
veteran$treat <- as.numeric(veteran$trt) - 1

result <- df_counting(
  df = veteran,
  tte.name = "time",
  event.name = "status",
  treat.name = "treat"
)

# Check consistency of test statistics
check_results(result)

# Store results without printing
stats_comparison <- check_results(result, verbose = FALSE)
print(stats_comparison)


# Simple example with constructed data
dfcount_example <- list(
  lr = 2.5,
  sig2_lr = 1.0,
  z.score = 2.5,
  logrank_results = list(chisq = 6.25)
)
check_results(dfcount_example)
```

---

ci_cox *Confidence interval for Cox model estimate*

Description

Calculates the confidence interval for a Cox model log hazard ratio estimate.

Usage

```
ci_cox(bhat, sig_bhat, alpha = 0.05, verbose = FALSE)
```

Arguments

bhat	Estimated log hazard ratio.
sig_bhat	Standard error of the estimate.
alpha	Significance level (default: 0.05).
verbose	Logical; if TRUE, prints interval.

Value

Data frame with log hazard ratio, standard error, hazard ratio, lower and upper confidence limits.

---

count_weighted *Weighted counting process*

Description

Computes the weighted count of events up to a specified time.

Usage

```
count_weighted(x, y, w = rep(1, length(y)))
```

Arguments

x	Time point.
y	Vector of event/censoring times.
w	Weights (default 1).

Value

Weighted count of events up to time x.

---

cox_rhogamma *Weighted Cox Model with Rho-Gamma Weights*

Description

Fits a weighted Cox proportional hazards model using flexible time-dependent weights (e.g., Fleming-Harrington, Magirr-Burman). Supports resampling-based inference for variance estimation and bias correction.

Usage

```
cox_rhogamma(
  dfcount,
  scheme = "fh",
  scheme_params = list(rho = 0, gamma = 0.5),
  draws = 0,
  alpha = 0.05,
  verbose = FALSE,
  lr.digits = 4
)
```

Arguments

`dfcount`	List; output from `df_counting` containing counting process data including time, delta, z, w_hat, survP_all, survG_all, etc.
`scheme`	Character; weighting scheme. See `wt.rg.S` for options. Default: "fh".
`scheme_params`	List; parameters for the selected scheme. Default: `list(rho = 0, gamma = 0.5)`. Required parameters depend on scheme: • "fh": `rho`, `gamma` • "MB": `mb_tstar` • "custom_time": `t.tau`, `w0.tau`, `w1.tau`
`draws`	Integer; number of resampling draws for variance estimation and bias correction. If 0, only asymptotic inference is performed. Default: 0.
`alpha`	Numeric; significance level for confidence intervals. Default: 0.05.
`verbose`	Logical; whether to print detailed output. Default: FALSE.
`lr.digits`	Integer; number of decimal places for formatted output. Default: 4.

Details

This function solves the weighted Cox partial likelihood score equation: U(beta) = sum_i w_i K_i (dN_0/Y_0 - dN_1/Y_1) = 0

where K_i are time-dependent weights and Y_j, dN_j are risk sets and event counts for group j.

When `draws > 0`, the function performs resampling to:

- Estimate finite-sample variance (more accurate than asymptotic)
- Compute bias correction for $\hat{\beta}$
- Provide improved confidence intervals for small samples

The score test at $\beta = 0$ corresponds to the weighted log-rank test.

Value

A list containing:

fit List with fitted model components:

- `bhat`: Estimated log hazard ratio

- `sig_bhat_asy`: Asymptotic standard error
- `u.zero`: Score statistic at beta=0 (log-rank)
- `z.score`: Standardized score statistic
- `sig2_score`: Variance of score statistic
- `wt_rg`: Vector of time-dependent weights
- `bhat_debiased`: Bias-corrected estimate (if draws > 0)
- `sig_bhat_star`: Resampling-based standard error (if draws > 0)

hr_ci_asy Data frame with asymptotic HR and CI

hr_ci_star Data frame with resampling-based HR and CI (if draws > 0)

cox_text_asy Formatted string with HR and asymptotic CI

cox_text_star Formatted string with HR and resampling CI (if draws > 0)

z.score_debiased Bias-corrected score statistic (if draws > 0)

zlogrank_text Formatted log-rank test result

Note

The treatment variable in `dfcount` must be coded as 0=control, 1=experimental.

References

Magirr, D. and Burman, C. F. (2019). Modestly weighted logrank tests. Statistics in Medicine, 38(20), 3782-3790.

See Also

`df_counting` for preprocessing `wt.rg.S` for weighting schemes `cox_score_rhogamma` for score function

Other survival_analysis: `KM_diff()`, `df_counting()`, `wt.rg.S()`

Other weighted_tests: `df_counting()`, `wt.rg.S()`

Examples

```
# First get counting process data
library(survival)
str(veteran)
veteran$treat <- as.numeric(veteran$trt) - 1

dfcount <- df_counting(
  df = veteran,
  tte.name = "time",
  event.name = "status",
  treat.name = "treat"
)

# Fit weighted Cox model with FH(0,0.5) weights
fit <- cox_rhogamma(
  dfcount = dfcount,
```

```
  scheme = "fh",
  scheme_params = list(rho = 0, gamma = 0.5),
  draws = 1000,
  verbose = TRUE
)

print(fit$cox_text_star)  # Resampling-based CI
print(fit$zlogrank_text)  # Weighted log-rank test

# Compare asymptotic and resampling CIs
print(fit$hr_ci_asy)
print(fit$hr_ci_star)
```

`cox_rhogamma_resample` *Resampling for Weighted Cox Model (rho, gamma)*

Description

Performs resampling to estimate uncertainty for the weighted Cox model (rho, gamma).

Usage

```
cox_rhogamma_resample(
  fit_rhogamma,
  i_bhat,
  K_wt_rg,
  i_zero,
  K_zero,
  G1.draws = NULL,
  G0.draws = NULL,
  draws = 100,
  seedstart = 8316951
)
```

Arguments

`fit_rhogamma`	List with fitted Cox model results.
`i_bhat`	Information at estimated beta.
`K_wt_rg`	Weights at estimated beta.
`i_zero`	Information at beta=0.
`K_zero`	Weights at beta=0.
`G1.draws`	Optional: pre-generated random draws for groups.
`G0.draws`	Optional: pre-generated random draws for groups.
`draws`	Number of resampling iterations (default: 100).
`seedstart`	Random seed for reproducibility (default: 8316951).

Value

List with resampling results (score, beta, standard error, etc.).

`cox_score_rhogamma` *Cox score with rho-gamma weights*

Description

Calculates the Cox score statistic for weighted log-rank tests.

Usage

```
cox_score_rhogamma(
  beta,
  time,
  delta,
  z,
  w_hat = rep(1, length(time)),
  wt_rg = rep(1, length(time)),
  score_only = TRUE
)
```

Arguments

`beta`	Log hazard ratio parameter.
`time`	Numeric vector of event times.
`delta`	Numeric vector of event indicators.
`z`	Numeric vector of group indicators.
`w_hat`	Numeric vector of weights.
`wt_rg`	Numeric vector of rho-gamma weights.
`score_only`	Logical; if TRUE, returns only the score.

Value

Numeric value of the score or a list with additional results.

`create_baseline_table` *Create Baseline Characteristics Table by Treatment Arm*

Description

This function generates publication-ready baseline characteristic tables commonly used in clinical trials and observational studies. It calculates summary statistics, p-values, and standardized mean differences for continuous, categorical, and binary variables.

Usage

```
create_baseline_table(
  data,
  treat_var = "treat",
  vars_continuous = NULL,
  vars_categorical = NULL,
  vars_binary = NULL,
  var_labels = NULL,
  digits = 1,
  show_pvalue = TRUE,
  show_smd = TRUE,
  show_missing = TRUE
)
```

Arguments

`data`	Data frame containing baseline variables
`treat_var`	Name of treatment variable (default: "treat")
`vars_continuous`	Character vector of continuous variable names
`vars_categorical`	Character vector of categorical variable names
`vars_binary`	Character vector of binary variable names
`var_labels`	Named vector for variable labels (e.g., c(age = "Age (years)"))
`digits`	Number of decimal places for continuous variables (default: 1)
`show_pvalue`	Logical, whether to show p-values (default: TRUE)
`show_smd`	Logical, whether to show standardized mean differences (default: TRUE)
`show_missing`	Logical, whether to show missing data counts (default: TRUE)

Value

A gt table object (if gt package is available) or data frame

Examples

```
# Create sample data
set.seed(123)
n <- 500
sample_data <- data.frame(
  treat = rbinom(n, 1, 0.5),
  age = rnorm(n, mean = 55, sd = 10),
  stage = sample(c("I", "II", "III", "IV"), n, replace = TRUE),
  sex = rbinom(n, 1, 0.45),
  smoking = rbinom(n, 1, 0.3)
)

# Create table
table <- create_baseline_table(
  data = sample_data,
  treat_var = "treat",
  vars_continuous = "age",
  vars_categorical = "stage",
  vars_binary = c("sex", "smoking"),
  var_labels = c(
    age = "Age (years)",
    stage = "Disease Stage",
    sex = "Female",
    smoking = "Current Smoker"
  )
)
```

---

cumulative_rmst_bands *Cumulative RMST bands for survival curves*

Description

Calculates cumulative Restricted Mean Survival Time (RMST) and confidence bands for survival curves using resampling. Optionally plots the cumulative RMST curve, pointwise confidence intervals, and simultaneous confidence bands.

Usage

```
cumulative_rmst_bands(
  df,
  fit,
  tte.name,
  event.name,
  treat.name,
  weight.name = NULL,
  draws_sb = 1000,
```

```
  xlab = "months",
  ylim_pad = 0.5,
  rmst_max_legend = "left",
  rmst_max_cex = 0.7,
  plot = TRUE
)
```

Arguments

df	Data frame containing survival data.
fit	Survival fit object (output from KM_diff).
tte.name	Name of time-to-event variable in df.
event.name	Name of event indicator variable in df.
treat.name	Name of treatment group variable in df.
weight.name	Optional name of weights variable in df.
draws_sb	Number of resampling draws for simultaneous bands (default: 1000).
xlab	Label for x-axis (default: "months").
ylim_pad	Padding for y-axis limits (default: 0.5).
rmst_max_legend	Position for RMST legend (default: "left").
rmst_max_cex	Text size for RMST legend (default: 0.7).
plot	Logical; if TRUE, plot the results. Default is TRUE.

Value

A list with elements:

at_points	Time points used for RMST calculation
rmst_time	Cumulative RMST estimates
sig2_rmst_time	Variance of RMST estimates
rmst_time_lower	Pointwise lower confidence interval
rmst_time_upper	Pointwise upper confidence interval
rmst_maxtau_ci	RMST and CI at maximum time
rmst_text	Text summary for legend
c_alpha_band	Critical value for simultaneous band
rmst_time_sb_lower	Simultaneous band lower bound
rmst_time_sb_upper	Simultaneous band upper bound

df_counting *Weighted and Stratified Survival Analysis*

Description

Performs comprehensive weighted and/or stratified survival analysis, including Cox proportional hazards model, logrank/Fleming-Harrington tests, and calculation of risk/event sets, Kaplan-Meier curves, quantiles, and variance estimates.

Usage

```
df_counting(
  df,
  tte.name = "tte",
  event.name = "event",
  treat.name = "treat",
  weight.name = NULL,
  strata.name = NULL,
  arms = c("treat", "control"),
  time.zero = 0,
  tpoints.add = c(0),
  by.risk = 6,
  time.zero.label = 0,
  risk.add = NULL,
  get.cox = TRUE,
  cox.digits = 2,
  lr.digits = 2,
  cox.eps = 0.001,
  lr.eps = 0.001,
  verbose = FALSE,
  qprob = 0.5,
  scheme = "fh",
  scheme_params = list(rho = 0, gamma = 0),
  conf_level = 0.95,
  check.KM = TRUE,
  check.seKM = FALSE,
  draws = 0,
  seedstart = 8316951,
  stop.onerror = FALSE,
  censoring_allmarks = TRUE
)
```

Arguments

df	Data frame containing survival data.
tte.name	Character; name of the time-to-event column in df.

`event.name` Character; name of the event indicator column in `df` (1=event, 0=censored).

`treat.name` Character; name of the treatment/group column in `df` (must be coded as 0=control, 1=experimental).

`weight.name` Character or NULL; name of the weights column in `df`. If NULL, equal weights are used.

`strata.name` Character or NULL; name of the strata column in `df` for stratified analysis.

`arms` Character vector of length 2; group labels. Default: `c("treat","control")`.

`time.zero` Numeric; time value to use as zero. Default: 0.

`tpoints.add` Numeric vector; additional time points to include in calculations. Default: `c(0)`.

`by.risk` Numeric; interval for risk set time points. Default: 6.

`time.zero.label` Numeric; label for time zero in output. Default: 0.0.

`risk.add` Numeric vector or NULL; additional specific risk points to include.

`get.cox` Logical; whether to fit Cox proportional hazards model. Default: TRUE.

`cox.digits` Integer; number of decimal places for Cox output formatting. Default: 2.

`lr.digits` Integer; number of decimal places for logrank output formatting. Default: 2.

`cox.eps` Numeric; threshold for Cox p-value formatting (values below shown as "<eps"). Default: 0.001.

`lr.eps` Numeric; threshold for logrank p-value formatting. Default: 0.001.

`verbose` Logical; whether to print warnings and diagnostic messages. Default: FALSE.

`qprob` Numeric in (0,1); quantile probability for KM quantile table. Default: 0.5 (median).

`scheme` Character; weighting scheme for logrank/Fleming-Harrington test. Options: "fh", "schemper", "XO", "MB", "custom_time", "fh_exp1", "fh_exp2". Default: "fh".

`scheme_params` List; parameters for the selected weighting scheme. Default: `list(rho = 0, gamma = 0)`.

- For "fh": `rho` and `gamma` (Fleming-Harrington parameters)
- For "MB": `mb_tstar` (cutoff time)
- For "custom_time": `t.tau`, `w0.tau`, `w1.tau`

`conf_level` Numeric in (0,1); confidence level for quantile intervals. Default: 0.95.

`check.KM` Logical; whether to check KM curve validity against `survival::survfit`. Default: TRUE.

`check.seKM` Logical; whether to check KM standard error estimates. Default: FALSE.

`draws` Integer; number of draws for resampling-based variance estimation. Default: 0 (no resampling).

`seedstart` Integer; random seed for reproducible resampling. Default: 8316951.

`stop.onerror` Logical; whether to stop execution on errors (TRUE) or issue warnings (FALSE). Default: FALSE.

`censoring_allmarks` Logical; if FALSE, removes events from censored time points. Default: TRUE.

Details

This function implements a comprehensive survival analysis framework supporting:

- Weighted observations via `weight.name`
- Stratified analysis via `strata.name`
- Multiple weighting schemes for log-rank tests
- Resampling-based variance estimation
- Automatic validation against `survival` package results

The function performs time-fixing using `survival::aeqSurv` to handle tied event times. For stratified analyses, stratum-specific estimates are computed and combined appropriately.

Value

A list with the following components:

cox_results List with Cox model results including hazard ratio, confidence interval, p-value, and formatted text

logrank_results List with log-rank test chi-square statistic, p-value, and formatted text

z.score Standardized weighted log-rank test statistic

at_points Vector of all time points used in calculations

surv0, surv1 Kaplan-Meier survival estimates for control and treatment groups

sig2_surv0, sig2_surv1 Variance estimates for survival curves

survP Pooled survival estimates

survG Censoring distribution estimates

quantile_results Data frame with median survival and confidence intervals by group

lr, sig2_lr Weighted log-rank statistic and its variance

riskpoints0, riskpoints1 Risk set counts at specified time points

z.score_stratified Stratified z-score (if stratified analysis)

Weighting Schemes

fh Fleming-Harrington: w(t) = S(t)^rho * (1-S(t))^gamma

MB Magirr-Burman: w(t) = 1/max(S(t), S(t*))

schemper Schemper: w(t) = S(t)/G(t) where G is the censoring distribution

XO Xu-O'Quigley: w(t) = S(t)/Y(t) where Y is risk set size

References

Fleming, T. R. and Harrington, D. P. (1991). Counting Processes and Survival Analysis. Wiley.

Magirr, D. and Burman, C. F. (2019). Modestly weighted logrank tests. Statistics in Medicine, 38(20), 3782-3790.

See Also

coxph, survdiff, survfit cox_rhogamma for weighted Cox models KM_diff for Kaplan-Meier differences

Other survival_analysis: KM_diff(), cox_rhogamma(), wt.rg.S()

Other weighted_tests: cox_rhogamma(), wt.rg.S()

Examples

```
# Basic survival analysis
library(survival)
str(veteran)
veteran$treat <- as.numeric(veteran$trt) - 1

result <- df_counting(
  df = veteran,
  tte.name = "time",
  event.name = "status",
  treat.name = "treat",
  arms = c("Treatment", "Control")
)

# Print results
print(result$cox_results$cox_text)
print(result$zlogrank_text)

# Fleming-Harrington (0,1) weights (emphasizing late differences)
result_fh <- df_counting(
  df = veteran,
  tte.name = "time",
  event.name = "status",
  treat.name = "treat",
  scheme = "fh",
  scheme_params = list(rho = 0, gamma = 1)
)

# Stratified analysis
result_strat <- df_counting(
  df = veteran,
  tte.name = "time",
  event.name = "status",
  treat.name = "treat",
  strata.name = "celltype"
)
```

`extract_and_calc_weights` *Extract and calculate weights for multiple schemes*

Description

Extracts and calculates weights for multiple schemes and returns a combined data frame.

Usage

```
extract_and_calc_weights(atpoints, S.pool, weights_spec_list)
```

Arguments

atpoints	Numeric vector of time points.
S.pool	Numeric vector of pooled survival probabilities.
weights_spec_list	List of weighting scheme specifications.

Value

Data frame with weights for each scheme.

---

extract_group_data *Extract time, event, and weight data for a group*

Description

Extracts time, event, and weight vectors for a specified group.

Usage

```
extract_group_data(time, delta, wgt, z, group = 1)
```

Arguments

time	Numeric vector of times.
delta	Numeric vector of event indicators (1=event, 0=censored).
wgt	Numeric vector of weights.
z	Numeric vector of group indicators.
group	Value of group to extract (default 1).

Value

List with U (times), D (events), W (weights).

find_cox_root *Root-finding for Cox score function*

Description

Finds the root of the Cox model score equation for weighted log-rank statistics.

Usage

```
find_cox_root(time, delta, z, w_hat, wt_rg)
```

Arguments

time	Numeric vector of event times.
delta	Numeric vector of event indicators.
z	Numeric vector of group indicators.
w_hat	Numeric vector of weights.
wt_rg	Numeric vector of rho-gamma weights.

Value

List with root and additional information, or NA if not found.

format_pval *Format p-value for display*

Description

Formats a p-value for display, showing "<eps" for small values.

Usage

```
format_pval(pval, eps = 0.001, digits = 3)
```

Arguments

pval	Numeric p-value.
eps	Threshold for small p-values.
digits	Number of digits to display.

Value

Formatted p-value as character.

get_censoring_and_events *Get censoring and event times and their indices*

Description

Extracts censoring and event times and their indices for a group at specified time points.

Usage

```
get_censoring_and_events(time, delta, z, group, censoring_allmarks, at_points)
```

Arguments

time	Numeric vector of times.
delta	Numeric vector of event indicators.
z	Numeric vector of group indicators.
group	Value of group to extract.
censoring_allmarks	Logical; if FALSE, remove events from censored.
at_points	Numeric vector of time points.

Value

List with cens (censored times), ev (event times), idx_cens, idx_ev, idx_ev_full.

get_dfcounting *Get df_counting results*

Description

Wrapper for df_counting, safely executes and returns results for survival analysis.

Usage

```
get_dfcounting(...)
```

Arguments

...	Arguments passed to df_counting.

Value

Result from df_counting or NULL if error.

See Also

df_counting

---

get_event_risk_matrices *Event and Risk Matrices for Survival Analysis*

Description

Constructs matrices indicating event and risk status for each subject at specified time points.

Usage

```
get_event_risk_matrices(U, at.points)
```

Arguments

U	Vector of observed times (e.g., time-to-event).
at.points	Vector of time points at which to evaluate events and risk.

Value

A list with event and risk matrices.

---

get_riskpoints *Get risk set counts at specified risk points*

Description

Returns risk set counts at specified risk points.

Usage

```
get_riskpoints(ybar, risk_points, at_points)
```

Arguments

ybar	Numeric vector of risk set counts.
risk_points	Numeric vector of risk points.
at_points	Numeric vector of time points.

Value

Numeric vector of risk set counts at risk points.

get_validated_weights *Get validated weights for a data frame*

Description

Validates and returns weights for a data frame according to specified schemes.

Usage

```
get_validated_weights(
  df_weights,
  scheme = "fh",
  scheme_params = list(rho = 0, gamma = 0),
  details = FALSE
)
```

Arguments

df_weights	Data frame containing weights and related data.
scheme	Character string specifying the weighting scheme.
scheme_params	List of parameters for the scheme.
details	Logical; if TRUE, returns detailed output.

Value

Numeric vector or list of validated weights.

get_weights *Get weights for a weighting scheme*

Description

Calculates weights for a specified scheme at given time points.

Usage

```
get_weights(scheme, scheme_params, S.pool, tpoints)
```

Arguments

scheme	Character string specifying the weighting scheme.
scheme_params	List of parameters for the scheme.
S.pool	Numeric vector of pooled survival probabilities.
tpoints	Numeric vector of time points.

Value

Numeric vector of weights.

`KM_diff` *Kaplan-Meier Difference Between Groups*

Description

Calculates the difference in Kaplan-Meier curves between two groups, with confidence intervals and optional resampling-based simultaneous confidence bands.

Usage

```
KM_diff(
  df,
  tte.name = "tte",
  event.name = "event",
  treat.name = "treat",
  weight.name = NULL,
  at_points = sort(df[[tte.name]]),
  alpha = 0.05,
  seedstart = 8316951,
  draws = 0,
  risk.points = NULL,
  draws.band = 0,
  tau.seq = 0.25,
  qtau = 0.025,
  show_resamples = TRUE,
  modify_tau = FALSE
)
```

Arguments

`df`	Data frame containing survival data.
`tte.name`	Character; name of time-to-event variable in `df`.
`event.name`	Character; name of event indicator variable in `df` (1=event, 0=censored).
`treat.name`	Character; name of treatment group variable in `df` (0=control, 1=treatment).
`weight.name`	Character or NULL; name of weights variable in `df`.
`at_points`	Numeric vector; time points for calculation. Default: sorted unique event times.
`alpha`	Numeric; significance level for confidence intervals. Default: 0.05.
`seedstart`	Integer; random seed for reproducibility. Default: 8316951.
`draws`	Integer; number of draws for pointwise variance estimation. Default: 0.
`risk.points`	Numeric vector; time points for risk table display.

draws.band	Integer; number of draws for simultaneous confidence bands. Default: 0.
tau.seq	Numeric; step size for tau sequence when modify_tau=TRUE. Default: 0.25.
qtau	Numeric; quantile for tau range restriction. Default: 0.025.
show_resamples	Logical; whether to plot resampled curves. Default: TRUE.
modify_tau	Logical; whether to restrict time range for simultaneous bands. Default: FALSE.

Details

This function computes the difference in Kaplan-Meier survival curves, delta(t) = S_1(t) - S_0(t),

along with variance estimates and confidence intervals.

When draws.band > 0, simultaneous confidence bands are constructed using the supremum distribution of the standardized difference process. These bands maintain the specified coverage probability across all time points simultaneously.

The variance is estimated using Greenwood's formula for unweighted data, or resampling-based methods when draws > 0.

Value

A list containing:

at_points Time points used in calculations

surv0, surv1 Survival estimates for control and treatment groups

sig2_surv0, sig2_surv1 Variance estimates for survival curves

dhat Survival difference (S1 - S0) at each time point

sig2_dhat Variance of survival difference

lower, upper Pointwise confidence limits (1 - alpha/2)

sb_lower, sb_upper Simultaneous band limits (if draws.band > 0)

c_alpha_band Critical value for simultaneous band (if draws.band > 0)

dhat_star Matrix of resampled differences (if draws.band > 0)

Zdhat_star Standardized resampled differences (if draws.band > 0)

Confidence Intervals vs Bands

- Pointwise CIs (lower, upper): Cover the true difference at each time point with probability 1-alpha
- Simultaneous bands (sb_lower, sb_upper): Cover the entire difference curve with probability 1-alpha

Note

Treatment must be coded as 0=control, 1=experimental. Event must be binary (0/1).

See Also

df_counting for full survival analysis plotKM.band_subgroups for visualization cumulative_rmst_bands for RMST analysis

Other survival_analysis: cox_rhogamma(), df_counting(), wt.rg.S()

Examples

```
library(survival)
str(veteran)
veteran$treat <- as.numeric(veteran$trt) - 1

# Basic KM difference
result <- KM_diff(
  df = veteran,
  tte.name = "time",
  event.name = "status",
  treat.name = "treat"
)

# Plot the difference
plot(result$at_points, result$dhat, type = "s",
     xlab = "Time", ylab = "Survival Difference")

# With simultaneous confidence bands
result_band <- KM_diff(
  df = veteran,
  tte.name = "time",
  event.name = "status",
  treat.name = "treat",
  draws.band = 1000,
  modify_tau = TRUE
)
```

KM_estimates *Kaplan-Meier Survival Estimates and Variance*

Description

Computes Kaplan-Meier survival estimates and their variances given risk and event counts.

Usage

```
KM_estimates(ybar, nbar, sig2w_multiplier = NULL)
```

Arguments

ybar	Vector of risk set sizes at each time point.
nbar	Vector of event counts at each time point.
sig2w_multiplier	Optional vector for variance calculation. If NULL, calculated internally.

Value

List with survival estimates and variances.

KM_plot_2sample_weighted_counting

Plot Weighted Kaplan-Meier Curves for Two Samples (Counting Process Format)

Description

Plots Kaplan-Meier survival curves for two groups using precomputed risk/event counts and survival estimates. Optionally displays confidence intervals, risk tables, median survival annotations, and statistical test results.

Usage

```
KM_plot_2sample_weighted_counting(
  dfcount,
  show.cox = TRUE,
  cox.cex = 0.725,
  show.logrank = FALSE,
  logrank.cex = 0.725,
  cox.eps = 0.001,
  lr.eps = 0.001,
  show_arm_legend = TRUE,
  arms = c("treat", "control"),
  put.legend.arms = "left",
  stop.onerror = TRUE,
  check.KM = TRUE,
  put.legend.cox = "topright",
  put.legend.lr = "topleft",
  lr.digits = 2,
  cox.digits = 2,
  tpoints.add = c(0),
  by.risk = NULL,
  Xlab = "time",
  Ylab = "proportion surviving",
  col.0 = "black",
  col.1 = "blue",
```

```
  show.med = TRUE,
  med.digits = 2,
  med.font = 4,
  conf.int = FALSE,
  conf_level = 0.95,
  choose_ylim = FALSE,
  arm.cex = 0.7,
  quant = 0.5,
  med.cex = 0.725,
  ymed.offset = 0.1,
  xmed.fraction = 0.8,
  risk.cex = 0.725,
  ltys = c(1, 1),
  lwds = c(1, 1),
  censor.mark.all = TRUE,
  censor.cex = 0.5,
  show.ticks = TRUE,
  risk.set = TRUE,
  ymin = 0,
  ymax = 1,
  ymin.del = 0.035,
  ymin2 = NULL,
  risk_offset = 0.125,
  risk_delta = 0.05,
  y.risk0 = NULL,
  show.Y.axis = TRUE,
  cex_Yaxis = 1,
  y.risk1 = NULL,
  add.segment = FALSE,
  risk.add = NULL,
  xmin = 0,
  xmax = NULL,
  x.truncate = NULL,
  time.zero = 0,
  prob.points = NULL
)
```

Arguments

dfcount	List containing precomputed survival data.
show.cox	Logical; show Cox model results.
cox.cex	Numeric; text size for Cox annotation.
show.logrank	Logical; show log-rank test results.
logrank.cex	Numeric; text size for log-rank annotation.
cox.eps	Numeric; small values for Cox calculations.
lr.eps	Numeric; small values for log-rank calculations.

show_arm_legend	Logical; show arm legend.
arms	Character vector of arm labels.
put.legend.arms	Character; legend positions.
stop.onerror	Logical; stop on KM curve errors.
check.KM	Logical; check KM curve validity.
put.legend.cox	Character; legend positions.
put.legend.lr	Character; legend positions.
lr.digits	Integer; digits for test results.
cox.digits	Integer; digits for test results.
tpoints.add	Numeric vector; additional time points to include (default: c(0)).
by.risk	Numeric; interval for risk table time points.
Xlab	Character; axis labels.
Ylab	Character; axis labels.
col.0	Color for control curve.
col.1	Color for treatment curve.
show.med	Logical; annotate median survival.
med.digits	Median annotation settings.
med.font	Median annotation settings.
conf.int	Logical; plot confidence intervals.
conf_level	Numeric; confidence level for intervals.
choose_ylim	Logical; auto-select y-axis limits.
arm.cex	Numeric; text size for arm legend.
quant	Numeric; quantile for annotation.
med.cex	Median annotation settings.
ymed.offset	Median annotation settings.
xmed.fraction	Median annotation settings.
risk.cex	Numeric; text size for risk table.
ltys	Integer; line types for curves.
lwds	Integer; line widths for curves.
censor.mark.all	Logical; mark all censored times.
censor.cex	Numeric; size of censor marks.
show.ticks	Logical; show axis ticks.
risk.set	Logical; display risk table.
ymin	Additional graphical and calculation parameters.
ymax	Additional graphical and calculation parameters.

ymin.del	Additional graphical and calculation parameters.
ymin2	Additional graphical and calculation parameters.
risk_offset	Additional graphical and calculation parameters.
risk_delta	Additional graphical and calculation parameters.
y.risk0	Additional graphical and calculation parameters.
show.Y.axis	Additional graphical and calculation parameters.
cex_Yaxis	Additional graphical and calculation parameters.
y.risk1	Additional graphical and calculation parameters.
add.segment	Additional graphical and calculation parameters.
risk.add	Additional graphical and calculation parameters.
xmin	Additional graphical and calculation parameters.
xmax	Additional graphical and calculation parameters.
x.truncate	Additional graphical and calculation parameters.
time.zero	Numeric; time zero value for risk table alignment (default: 0).
prob.points	Numeric vector; probability points for additional annotations (default: NULL).

Value

Invisibly returns NULL. Used for plotting side effects.

`km_quantile` *Kaplan-Meier quantile and confidence interval*

Description

Calculates the quantile and confidence interval for a Kaplan-Meier curve.

Usage

```
km_quantile(
  time_points,
  survival_probs,
  se_probs = NULL,
  qprob = 0.5,
  type = c("midpoint", "min"),
  conf_level = 0.95
)
```

Arguments

time_points	Vector of time points.
survival_probs	Vector of survival probabilities.
se_probs	Standard errors of survival probabilities.
qprob	Quantile probability (default 0.5).
type	Calculation type (midpoint or min).
conf_level	Confidence level (default 0.95).

Value

List with quantile and confidence interval.

---

km_quantile_table *Table of KM quantiles for two groups*

Description

Returns a data frame of quantiles and confidence intervals for two groups.

Usage

```
km_quantile_table(
  time_points,
  surv0,
  se0,
  surv1,
  se1,
  arms = c("treat", "control"),
  qprob = 0.5,
  type = c("midpoint", "min"),
  conf_level = 0.95
)
```

Arguments

time_points	Vector of time points.
surv0	Survival probabilities for group 0.
se0	Standard errors for group 0.
surv1	Survival probabilities for group 1.
se1	Standard errors for group 1.
arms	Group labels.
qprob	Quantile probability.
type	Calculation type.
conf_level	Confidence level.

Value

Data frame of quantiles and CIs for each group.

---

kmq_calculations *Kaplan-Meier quantile calculation*

Description

Calculates the quantile time for a Kaplan-Meier curve.

Usage

```
kmq_calculations(time_points, survival_probs, qprob = 0.5, type = "midpoint")
```

Arguments

time_points	Vector of time points.
survival_probs	Vector of survival probabilities.
qprob	Quantile probability (default 0.5).
type	Calculation type (midpoint or min).

Value

Estimated quantile time.

N_rhogamma *Weighted event count for Cox model*

Description

Calculates the weighted event count for the Cox model with rho-gamma weights.

Usage

```
N_rhogamma(x, error, delta, weight = 1)
```

Arguments

x	Numeric vector of time points.
error	Numeric vector of error terms.
delta	Numeric vector of event indicators.
weight	Numeric vector of weights (default: 1).

Value

Numeric value of weighted event count.

plot_km *Plot Kaplan-Meier curves*

Description

Plots Kaplan-Meier survival curves for groups in the data.

Usage

```
plot_km(df, tte.name, event.name, treat.name, weights = NULL, ...)
```

Arguments

df	Data frame containing survival data.
tte.name	Name of time-to-event column.
event.name	Name of event indicator column.
treat.name	Name of treatment/group column.
weights	Optional; name of weights column.
...	Additional arguments passed to plot().

Value

Kaplan-Meier fit object (invisible).

plot_km_confint_polygon *Plot confidence interval polygon for KM curve*

Description

Plots a shaded polygon representing the confidence interval for a Kaplan-Meier survival curve.

Usage

```
plot_km_confint_polygon(x, surv, se, conf_level, col)
```

Arguments

x	Numeric vector of time points.
surv	Numeric vector of survival probabilities.
se	Numeric vector of standard errors of survival probabilities.
conf_level	Numeric; confidence level for interval (default 0.95).
col	Color for the polygon.

Value

Invisibly returns NULL. Used for plotting side effects.

`plot_km_curves_counting` *Plot KM curves for two groups with optional confidence intervals and censoring marks*

Description

Plots Kaplan-Meier survival curves for two groups, with options for confidence intervals and censoring marks.

Usage

```
plot_km_curves_counting(
  at_points,
  S0.KM,
  idx0,
  idv0,
  S1.KM,
  idx1,
  idv1,
  col.0,
  col.1,
  ltys,
  lwds,
  Xlab,
  Ylab,
  ylim,
  xlim,
  show.ticks = FALSE,
  cens0 = NULL,
  risk.points,
  risk.points.label,
  cens1 = NULL,
  se0.KM = NULL,
  se1.KM = NULL,
  conf.int = FALSE,
  conf_level = 0.95,
  censor.cex = 1,
  time.zero = 0,
  tpoints.add = c(0),
  ...
)
```

Arguments

at_points	Numeric vector of time points for plotting.
S0.KM	Numeric vector of survival probabilities for group 0.
idx0	Indices for censoring in group 0.
idv0	Indices for events in group 0.
S1.KM	Numeric vector of survival probabilities for group 1.
idx1	Indices for censoring in group 1.
idv1	Indices for events in group 1.
col.0	Color for group 0.
col.1	Color for group 1.
ltys	Line types for groups.
lwds	Line widths for groups.
Xlab	X-axis label.
Ylab	Y-axis label.
ylim	Y-axis limits.
xlim	X-axis limits.
show.ticks	Logical; show censoring marks (default FALSE).
cens0	Numeric vector of censoring times for group 0.
risk.points	Numeric vector of risk time points.
risk.points.label	Character vector of labels for risk time points.
cens1	Numeric vector of censoring times for group 1.
se0.KM	Numeric vector of standard errors for group 0.
se1.KM	Numeric vector of standard errors for group 1.
conf.int	Logical; show confidence intervals (default FALSE).
conf_level	Confidence level (default 0.95).
censor.cex	Numeric; censoring mark size (default 1.0).
time.zero	Numeric; time zero value for risk table alignment (default 0).
tpoints.add	Numeric vector; additional time points to include (default c(0)).
...	Additional arguments to plot.

Value

Invisibly returns NULL. Used for plotting side effects.

`plot_weight_schemes` *Plot weight schemes for survival analysis*

Description

Plots the weights for different schemes over time using ggplot2.

Usage

```
plot_weight_schemes(
  dfcount,
  tte.name = "time_months",
  event.name = "status",
  treat.name = "treat",
  arms = c("treat", "control"),
  weights_spec_list = list(`MB(12)` = list(scheme = "MB", mb_tstar = 12), `MB(6)` =
   list(scheme = "MB", mb_tstar = 6), `FH(0,1)` = list(scheme = "fh", rho = 0, gamma =
    1), `FH(0.5,0.5)` = list(scheme = "fh", rho = 0.5, gamma = 0.5), custom_time =
    list(scheme = "custom_time", t.tau = 25, w0.tau = 1, w1.tau = 1.5), FHexp2 =
    list(scheme = "fh_exp2")),
  custom_colors = c(`FH(0,1)` = "grey", FHexp2 = "black", `MB(12)` = "#1b9e77", `MB(6)` =
    "#d95f02", `FH(0.5,0.5)` = "#7570b3", custom_time = "green"),
  custom_sizes = c(`FH(0,1)` = 2, FHexp2 = 1, `MB(12)` = 1, `MB(6)` = 1, `FH(0.5,0.5)` =
    1, custom_time = 1),
  transform_fh = FALSE,
  rescheme_fhexp2 = TRUE
)
```

Arguments

`dfcount`	Data frame containing counting process results, including time points and survival probabilities.
`tte.name`	Name of the time-to-event variable (default: 'time_months').
`event.name`	Name of the event indicator variable (default: 'status').
`treat.name`	Name of the treatment group variable (default: 'treat').
`arms`	Character vector of group names (default: c('treat', 'control')).
`weights_spec_list`	List of weighting scheme specifications.
`custom_colors`	Named character vector of colors for each scheme.
`custom_sizes`	Named numeric vector of line sizes for each scheme.
`transform_fh`	Logical; whether to transform FH weights (default: FALSE).
`rescheme_fhexp2`	Logical; whether to rescheme FHexp2 and custom_time (default: TRUE).

Details

This function visualizes the weights used in various survival analysis schemes (e.g., FH, MB, custom) using ggplot2. Facets and colors are customizable.

Value

A ggplot object showing the weight schemes over time.

`plot_weighted_km` *Plot weighted Kaplan-Meier curves*

Description

Plots weighted Kaplan-Meier curves using a custom function.

Usage

```
plot_weighted_km(dfcount, ...)
```

Arguments

`dfcount`	Result object from df_counting.
`...`	Additional arguments passed to KM_plot_2sample_weighted_counting.

Value

None. Plots the curves.

`plotKM.band_subgroups` *Plot Kaplan-Meier Survival Difference Curves with Subgroups and Confidence Bands*

Description

Plots the difference in Kaplan-Meier survival curves between two groups, optionally including simultaneous confidence bands and subgroup curves. Also displays risk tables for the overall population and specified subgroups.

Usage

```
plotKM.band_subgroups(
  df,
  tte.name,
  event.name,
  treat.name,
  weight.name = NULL,
  sg_labels = NULL,
  ltype = "s",
  lty = 1,
  draws = 20,
  lwd = 2,
  sg_colors = NULL,
  color = "lightgrey",
  ymax.pad = 0.01,
  ymin.pad = -0.01,
  taus = c(-Inf, Inf),
  yseq_length = 5,
  cex_Yaxis = 0.8,
  risk_cex = 0.8,
  by.risk = 6,
  risk.add = NULL,
  xmax = NULL,
  ymin = NULL,
  ymax = NULL,
  ymin.del = 0.035,
  y.risk1 = NULL,
  y.risk2 = NULL,
  ymin2 = NULL,
  risk_offset = NULL,
  risk.pad = 0.01,
  risk_delta = 0.0275,
  tau_add = NULL,
  time.zero.pad = 0,
  time.zero.label = 0,
  xlabel = NULL,
  ylabel = NULL,
  Maxtau = NULL,
  seedstart = 8316951,
  ylim = NULL,
  draws.band = 20,
  qtau = 0.025,
  show_resamples = FALSE,
  modify_tau = FALSE
)
```

Arguments

df	Data frame containing survival data.
tte.name	Name of the time-to-event column.
event.name	Name of the event indicator column (0/1).
treat.name	Name of the treatment group column (0/1).
weight.name	Optional name of the weights column.
sg_labels	Character vector of subgroup definitions (as logical expressions).
ltype	Line type for curves (default: "s").
lty	Line style for curves (default: 1).
draws	Number of draws for resampling (default: 20).
lwd	Line width for curves (default: 2).
sg_colors	Colors for subgroup curves.
color	Color for confidence band polygon (default: "lightgrey").
ymax.pad	Padding for y-axis limits.
ymin.pad	Padding for y-axis limits.
taus	Vector for time truncation (default: c(-Inf, Inf)).
yseq_length	Number of y-axis ticks (default: 5).
cex_Yaxis	Text size for axis.
risk_cex	Text size for risk table.
by.risk	Interval for risk table time points (default: 6).
risk.add	Additional time points for risk table.
xmax	Additional graphical and calculation parameters.
ymin	Additional graphical and calculation parameters.
ymax	Additional graphical and calculation parameters.
ymin.del	Additional graphical and calculation parameters.
y.risk1	Additional graphical and calculation parameters.
y.risk2	Additional graphical and calculation parameters.
ymin2	Additional graphical and calculation parameters.
risk_offset	Additional graphical and calculation parameters.
risk.pad	Additional graphical and calculation parameters.
risk_delta	Additional graphical and calculation parameters.
tau_add	Additional graphical and calculation parameters.
time.zero.pad	Additional graphical and calculation parameters.
time.zero.label	Additional graphical and calculation parameters.
xlabel	Additional graphical and calculation parameters.
ylabel	Additional graphical and calculation parameters.

Maxtau	Additional graphical and calculation parameters.
seedstart	Additional graphical and calculation parameters.
ylim	Additional graphical and calculation parameters.
draws.band	Number of draws for simultaneous confidence bands (default: 20).
qtau	Quantile for time range in simultaneous bands (default: 0.025).
show_resamples	Logical; whether to plot resampled curves (default: FALSE).
modify_tau	Logical; restrict time range for bands.

Value

(Invisible) list containing KM_diff results, time points, subgroup curves, risk tables, and confidence intervals.

---

resampling_survival *Resampling Survival Curves for Confidence Bands*

Description

Performs resampling to generate survival curves for a group, used for constructing confidence bands.

Usage

```
resampling_survival(U, W, D, at.points, draws.band, surv, G_draws)
```

Arguments

U	Vector of observed times.
W	Vector of weights.
D	Vector of event indicators (0/1).
at.points	Vector of time points for evaluation.
draws.band	Number of resampling draws.
surv	Vector of survival estimates.
G_draws	Matrix of random draws for resampling.

Value

Matrix of resampled survival curves.

risk_weighted *Weighted risk set*

Description

Computes the weighted number at risk at a specified time.

Usage

```
risk_weighted(x, y, w = rep(1, length(y)))
```

Arguments

x	Time point.
y	Vector of event/censoring times.
w	Weights (default 1).

Value

Weighted number at risk at time x.

safe_run *Safe execution wrapper*

Description

Executes an R expression safely, returning NULL and printing an error message if an error occurs.

Usage

```
safe_run(expr)
```

Arguments

expr	An R expression to evaluate.

Value

The result of expr, or NULL if an error occurs.

score_calculation *Score calculation for weighted Cox model*

Description

Calculates the score and variance for the weighted Cox model.

Usage

```
score_calculation(ybar1, ybar0, dN1, dN0, wt_rg)
```

Arguments

ybar1	Numeric vector of event counts for group 1.
ybar0	Numeric vector of event counts for group 0.
dN1	Numeric vector of event increments for group 1.
dN0	Numeric vector of event increments for group 0.
wt_rg	Numeric vector of rho-gamma weights.

Value

List with score, variance, information, and weights.

validate_input *Validate required columns in a data frame*

Description

Checks that all required columns are present in a data frame.

Usage

```
validate_input(df, required_cols)
```

Arguments

df	Data frame to check.
required_cols	Character vector of required column names.

Value

NULL if all columns present, otherwise error.

`validate_scheme_params` *Validate weighting scheme parameters*

Description

Checks and validates the parameters for a given weighting scheme.

Usage

```
validate_scheme_params(scheme, scheme_params, S.pool)
```

Arguments

`scheme`	Character string specifying the weighting scheme.
`scheme_params`	List of parameters for the scheme.
`S.pool`	Numeric vector of pooled survival probabilities.

Value

Logical indicating if parameters are valid, or stops with error.

`wlr_cumulative` *Weighted log-rank cumulative statistics*

Description

Calculates cumulative weighted log-rank statistics for survival data.

Usage

```
wlr_cumulative(
  df_weights,
  scheme,
  scheme_params = list(rho = 0, gamma = 0),
  return_cumulative = FALSE
)
```

Arguments

`df_weights`	Data frame with weights and risk/event counts.
`scheme`	Weighting scheme.
`scheme_params`	List of scheme parameters.
`return_cumulative`	Logical; whether to return cumulative statistics.

Value

List with score and variance.

`wlr_dhat_estimates` *Weighted Log-Rank and Difference Estimate at a Specified Time*

Description

Computes the weighted log-rank statistic, its variance, the difference in survival at a specified time (`tzero`), the variance of the difference, their covariance, and correlation, using flexible time-dependent weights. The weighting scheme is selected via the `scheme` argument and is calculated using `wt.rg.S`.

Usage

```
wlr_dhat_estimates(
  dfcounting,
  scheme = "fh",
  scheme_params = list(rho = 0, gamma = 0),
  tzero = NULL
)
```

Arguments

`dfcounting`	List output from `df_counting` containing risk sets, event counts, and survival estimates.
`scheme`	Character string specifying weighting scheme. One of: "fh" (Fleming-Harrington), "schemper", "XO", "MB", "custom_time", or "custom_code".
`scheme_params`	Named list with numeric weighting parameters `rho` and `gamma` (used for "fh" and "custom_code" schemes).
`tzero`	Time point at which to evaluate the difference in survival (default: 24).

Details

The weighting scheme is selected via the `scheme` argument and calculated using `wt.rg.S`. Supports standard Fleming-Harrington, Schemper, XO (Xu & O'Quigley), MB (Maggir-Burman), custom time-based, and custom code weights.

Value

A list with elements:

`lr`	Weighted log-rank test statistic.
`sig2_lr`	Variance of the log-rank statistic.
`dhat`	Difference in survival at `tzero`.
`cov_wlr_dhat`	Covariance between log-rank and difference at `tzero`.

sig2_dhat	Variance of the difference at tzero.
cor_wlr_dhat	Correlation between log-rank and difference at tzero.

---

wt.rg.S *Compute Time-Dependent Weights for Survival Analysis*

Description

Calculates time-dependent weights for survival analysis according to various schemes (Fleming-Harrington, Schemper, XO, MB, custom).

Usage

```
wt.rg.S(
  S,
  scheme = c("fh", "schemper", "XO", "MB", "custom_time", "fh_exp1", "fh_exp2"),
  rho = NULL,
  gamma = NULL,
  Scensor = NULL,
  Ybar = NULL,
  tpoints = NULL,
  t.tau = NULL,
  w0.tau = 0,
  w1.tau = 1,
  mb_tstar = NULL,
  details = FALSE
)
```

Arguments

S	Numeric vector of survival probabilities at each time point.
scheme	Character; weighting scheme. One of: 'fh', 'schemper', 'XO', 'MB', 'custom_time', 'fh_exp1', 'fh_exp2'.
rho	Numeric; rho parameter for Fleming-Harrington weights. Required if scheme='fh'.
gamma	Numeric; gamma parameter for Fleming-Harrington weights. Required if scheme='fh'.
Scensor	Numeric vector; censoring KM curve for Schemper weights. Must have same length as S.
Ybar	Numeric vector; risk set sizes for XO weights. Must have same length as S.
tpoints	Numeric vector; time points corresponding to S. Required for MB and custom_time schemes.
t.tau	Numeric; cutoff time for custom_time weights.
w0.tau	Numeric; weight before t.tau for custom_time weights.
w1.tau	Numeric; weight after t.tau for custom_time weights.
mb_tstar	Numeric; cutoff time for Magirr-Burman weights.
details	Logical; if TRUE, returns list with weights and diagnostic info. Default: FALSE.

Details

This function implements several weighting schemes for weighted log-rank tests: See vignette for details This function implements several weighting schemes for weighted log-rank tests:

Fleming-Harrington (fh) w(t) = S(t-)^rho * (1-S(t-))^gamma. See vignette for common choices.

Schemper w(t) = S(t-)/G(t-) where G is the censoring distribution.

Xu-O'Quigley (XO) w(t) = S(t-)/Y(t) where Y is risk set size.

Magirr-Burman (MB) w(t) = 1/max(S(t-), S(t*)).

Custom time Step function with weight w_0 before t* and w_1 after t*.

Value

If `details=FALSE` (default): Numeric vector of weights with length equal to `S`.

If `details=TRUE`: List containing:

weights Numeric vector of calculated weights

S Input survival probabilities

S_left Left-continuous survival probabilities

scheme Scheme name

Note

All weights are calculated using left-continuous survival probabilities S(t-) to ensure consistency with counting process notation.

References

Fleming, T. R. and Harrington, D. P. (1991). Counting Processes and Survival Analysis. Wiley.

Magirr, D. and Burman, C. F. (2019). Modestly weighted logrank tests. Statistics in Medicine, 38(20), 3782-3790.

Schemper, M., Wakounig, S., and Heinze, G. (2009). The estimation of average hazard ratios by weighted Cox regression. Statistics in Medicine, 28(19), 2473-2489.

See Also

Other survival_analysis: `KM_diff()`, `cox_rhogamma()`, `df_counting()`

Other weighted_tests: `cox_rhogamma()`, `df_counting()`

Examples

```
# Generate example survival curve
time <- seq(0, 24, by = 0.5)
surv <- exp(-0.05 * time)  # Exponential survival

# Fleming-Harrington (0,1) weights
w_fh01 <- wt.rg.S(surv, scheme = "fh", rho = 0, gamma = 1, tpoints = time)
```

```
# Magirr-Burman weights
w_mb <- wt.rg.S(surv, scheme = "MB", mb_tstar = 12, tpoints = time)

# Standard log-rank (equal weights)
w_lr <- wt.rg.S(surv, scheme = "fh", rho = 0, gamma = 0)

# Plotting examples
plot(time, w_fh01, type = "l", main = "FH(0,1) Weights")
plot(time, w_mb, type = "l", main = "MB(12) Weights")

# Compare multiple schemes
plot(time, w_lr, type = "l", ylim = c(0, 2))
lines(time, w_fh01, col = "blue")
legend("topleft", c("Log-rank", "FH(0,1)"), col = 1:2, lty = 1)
```

Index

∗ **diagnostic_functions**
    check_results, 7
∗ **plotting_functions**
    KM_diff, 26
∗ **survival_analysis**
    cox_rhogamma, 9
    df_counting, 17
    KM_diff, 26
    wt.rg.S, 47
∗ **weighted_tests**
    cox_rhogamma, 9
    df_counting, 17
    wt.rg.S, 47

add_legends, 3
add_median_annotation, 4
add_risk_table, 5

calculate_risk_event_counts, 6
check_km_curve, 6
check_results, 7
ci_cox, 8
count_weighted, 9
cox_rhogamma, 9, *20*, *28*, *48*
cox_rhogamma_resample, 12
cox_score_rhogamma, *11*, 13
coxph, *20*
create_baseline_table, 14
cumulative_rmst_bands, 15, *28*

df_counting, *7, 8*, *10, 11*, 17, *23, 24*, *28*, *48*

extract_and_calc_weights, 20
extract_group_data, 21

find_cox_root, 22
format_pval, 22

get_censoring_and_events, 23
get_dfcounting, 23
get_event_risk_matrices, 24
get_riskpoints, 24
get_validated_weights, 25
get_weights, 25

KM_diff, *11*, *20*, 26, *48*
KM_estimates, 28
KM_plot_2sample_weighted_counting, 29
km_quantile, 32
km_quantile_table, 33
kmq_calculations, 34

N_rhogamma, 34

plot_km, 35
plot_km_confint_polygon, 35
plot_km_curves_counting, 36
plot_weight_schemes, 38
plot_weighted_km, 39
plotKM.band_subgroups, *28*, 39

resampling_survival, 42
risk_weighted, 43

safe_run, 43
score_calculation, 44
survdiff, *20*
survfit, *20*

validate_input, 44
validate_scheme_params, 45

wlr_cumulative, 45
wlr_dhat_estimates, 46
wt.rg.S, *10, 11*, *20*, *28*, 47