Package 'bunsen'

May 29, 2025

Title Marginal Survival Estimation with Covariate Adjustment

Version 0.1.0

Description
Provides an efficient and robust implementation for estimating marginal Hazard Ratio (HR) and Restricted Mean Survival Time (RMST) with covariate adjustment using Daniel et al. (2021) <doi:10.1002/bimj.201900297> and Karrison et al. (2018) <doi:10.1177/1740774518759281>.

Maintainer Xinlei Deng <xinlei.deng@novartis.com>

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.3.1

Depends R (>= 2.10)

LazyData true

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

LinkingTo Rcpp

Imports stats, survival, boot, clustermq, Rcpp

NeedsCompilation yes

Author Xinlei Deng [cre, aut] (ORCID: <https://orcid.org/0000-0001-8129-6007>),
Mark Baillie [aut] (ORCID: <https://orcid.org/0000-0002-5618-0667>),
Craig Wang [aut] (ORCID: <https://orcid.org/0000-0003-1804-2463>),
Dominic Magirr [aut],
Alex Przybylski [aut]

Repository CRAN

Date/Publication 2025-05-29 08:50:10 UTC

Contents

`calculate_statistics` *Estimate the marginal causal survival curves using potential outcome framework*

Description

Estimate the marginal causal survival curves for simulating time-to-event data in a discrete manner based on the methods from Daniel et al.(2020).

Usage

```
calculate_statistics(model, trt)
```

Arguments

`model`	A fitted coxph model. This should be a coxph event model or censoring model.
`trt`	Character. Name of the treatment assignment variable.

Details

If the study period for the original data is divided into discrete windows, defined by the event times in the original data, at time t0 = 0, everyone in the simulated data is still a survivor. S(x) is the estimated survival function. By the end of the window (0,t1], a proportion S(t1) still survives. The conditional probability of surviving the next window, (t1,t2], conditional on surviving the first window, is S(t2)/S(t1), and so on. This function returns the S(t2)/S(t1) in series.

Value

Two vectors containing the marginal causal survival curves for treatment arms (1 for treatment arm; 0 for control arm/placebo). Each number is the probability of the surviving the time window (t1,t2],... conditional on surviving the prior corresponding window.

References

Daniel R, Zhang J, Farewell D. Making apples from oranges: Comparing noncollapsible effect estimators and their standard errors after adjustment for different covariate sets. Biom J. 2021;63(3):528-557. doi:10.1002/bimj.201900297

Examples

```
library(survival)
data("oak")

cox_event <- coxph(Surv(OS, os.status) ~ trt + btmb + pdl1, data = oak)
calculate_statistics(model = cox_event, trt = "trt")
```

`calculate_trt_effect` *Calculate the marginal treatment effect using counterfactual simulations*

Description

This function is used to calculate the logHR using cox model after obtaining the counterfactural simulations for potential outcomes from simulate_counterfactuals.

Usage

```
calculate_trt_effect(sim_out_1d, sim_out_0d, sim_out_1c, sim_out_0c)
```

Arguments

`sim_out_1d`	List. A list from simulate_counterfactuals for cox_event in treatment group, e.g. the cox model using OS.
`sim_out_0d`	List. A list from simulate_counterfactuals for cox_event in control group, e.g. the cox model using OS.
`sim_out_1c`	List. A list from simulate_counterfactuals for cox_event in treatment group, e.g. the cox model using 1-OS.
`sim_out_0c`	List. A list from simulate_counterfactuals for cox_event in control group, e.g. the cox model using 1-OS.

Details

The event indicator from this function uses the equation 10', I(Y0<Y0_cens) or I(Y1<Y1_cens).

Value

The marginal beta (logHR)

References

Daniel R, Zhang J, Farewell D. Making apples from oranges: Comparing noncollapsible effect estimators and their standard errors after adjustment for different covariate sets. Biom J. 2021;63(3):528-557. doi:10.1002/bimj.201900297

Examples

```
library(survival)
data("oak")

cox_event <- coxph(Surv(OS, os.status) ~ trt + btmb + pdl1, data = oak)
#
cox_censor <- coxph(Surv(OS, 1 - os.status) ~ trt + btmb + pdl1, data = oak)
bh <- basehaz(cox_event, centered = FALSE)
bh_c <- basehaz(cox_censor, centered = FALSE)
s_condi <- calculate_statistics(model = cox_event, trt = "trt")
s_condi_c <- calculate_statistics(model = cox_censor, trt = "trt")
sim_out_1d <- simulate_counterfactuals(
bh = bh, surv_cond = s_condi$surv_cond1, cpp = FALSE, M = 1000)
sim_out_0d <- simulate_counterfactuals(
bh = bh, surv_cond = s_condi$surv_cond0, cpp = FALSE, M = 1000)
sim_out_1c <- simulate_counterfactuals(
bh = bh_c, surv_cond = s_condi_c$surv_cond1, cpp = FALSE, M = 1000)
sim_out_0c <- simulate_counterfactuals(
bh = bh_c, surv_cond = s_condi_c$surv_cond0, cpp = FALSE, M = 1000)

output <- calculate_trt_effect(sim_out_1d, sim_out_0d, sim_out_1c, sim_out_0c)
```

`clmqControl` *Control of marginal HR estimation via clustermq*

Description

Construct control structures for marginal HR estimation. Specifically, this is the control for parallel computation via clustermq. It provides the nested parallel computation via LSF (remote parallel computation within remote parallel computation) and local multiprocess within remote parallel computation.

Usage

```
clmqControl(
  memory = 1024 * 32,
  local_se = FALSE,
  clmq_se = FALSE,
  clmq_hr = TRUE,
  clmq_local = FALSE,
  n_jobs = 100,
  local_cores = 1
)
```

Arguments

memory	Numeric. Memory allocation for the remote workers.
local_se	Bool. True for calculating SE using local multiprocess in remote workers. This is only useful when clmq_se = TRUE.
clmq_se	Bool. True for using parallel computation (nested or local) via clustermq. This can be combined with local_se to calculate SE with nested parallel computation or local multiprocess. Nested parallel computation means double parallel computations – each worker will do a parallel computation for simulate_counterfactuals. False for calculating SE only in remote workers without nested parallel computation and local multiprocess.
clmq_hr	Bool. True for calculating point estimate (marginal HR) using parallel computation via clustermq.
clmq_local	Bool. True for calculating point estimate (marginal HR) using local multiprocess in remote workers. This is only useful when clmq_hr = TRUE.
n_jobs	Numeric. Number of remote workers via clustermq.
local_cores	Numeric. Number of cores or processes used in local multiprocess. This is only useful when local_se or clmq_local = TRUE.

Details

The control function provides options to set the memory of each remote node, number of cpus used by each remote node, and computation approach (whether or not use the nested parallel computation or local multiprocess with parallel computation).

Value

A list containing the control arguments.

Examples

```
## Not run:
#Don't run as it requires LSF scheduler

library(survival)
data("oak")

cox_event <- coxph(Surv(OS, os.status) ~ trt + btmb + pdl1, data = oak)

cox_censor <- coxph(Surv(OS, 1 - os.status) ~ trt + btmb + pdl1, data = oak)


get_marginal_effect(
  trt = "trt", cox_event = cox_event, cox_censor = cox_censor, SE = TRUE,
  M = 1000, n.boot = 10, data = oak, seed = 1, cpp = FALSE, control = clmqControl()
)



## End(Not run)
```

`get_marginal_effect` *Calculate the marginal treatment effects for hazard ratio (HR) adjusting covariates in clinical trials*

Description

Use the simulation approach from Daniel et al. (2020) to estimate the marginal HR when adjusting covariates. Standard error of marginal HR was estimated via the nonparametric bootstrap. The standard error of HR will converge with the increase of bootstrap. We suggest setting a large M for a more robust estimation. This function uses the parallel computation via remote LSF and local multiprocess. Additional feature also includes the C++ optimization that can speed up the calculation.

Usage

```
get_marginal_effect(
  trt,
  cox_event,
  cox_censor,
  data,
  M,
  SE = TRUE,
  seed = NULL,
  cpp = TRUE,
  n.boot = 1000,
  control = clmqControl(),
  verbose = TRUE
)
```

Arguments

`trt`	Character. Variable name of the treatment assignment. Only support two arm trial at the moment.
`cox_event`	Object. A coxph model using the survival time and survival status.
`cox_censor`	Object. A coxph model using the survival time and 1-survival status.
`data`	A data frame used for cox_event and cox_censor.
`M`	Numeric. The number of simulated counterfactual patients. Suggest to set a large number to get robust results, but this will be very time comsuming.
`SE`	Bool. True for estimating SE. False for not estimating SE.
`seed`	Numeric. Random seed for simulation.
`cpp`	Bool. True for using C++ optimization. False for not using C++ optimization. This requires cpp package installed.
`n.boot`	Numeric. Number of bootstrap used.

control	Named list. A list containing control parameters, including memory of remote workers, whether to use nested parallel computation or local multiprocess, number of remote workers/jobs, etc. See details of clmqControl.
verbose	Bool. Print status messages. Default: TRUE

Details

In clinical trials, adjusting covariates like prognostic factors in the main analysis can increase the precision resulting in smaller SE. However, adjusting covariates in nonlinear models changes the target estimands, e.g. from marginal treatment effects to conditional treatment effects. This function has implemented the methods of Daniel et al. (2020) to estimate the marginal treatment effects when adjusting covariates. Increasing the M - the number of bootstrap can significantly increase the computation time. Hence, we introduced C++ optimization and parallel computation to speed up the calculation. It provides nested parallel computation via LSF and remote parallel computation with local multiprocess.

Value

A vector containing the marginal beta (logHR), standard error, and 95% CI.

References

Daniel R, Zhang J, Farewell D. Making apples from oranges: Comparing noncollapsible effect estimators and their standard errors after adjustment for different covariate sets. Biom J. 2021;63(3):528-557. doi:10.1002/bimj.201900297

Examples

```
## Not run:
#Don't run as it requires LSF scheduler

library(survival)
data("oak")

cox_event <- coxph(Surv(OS, os.status) ~ trt + btmb + pdl1, data = oak)
#
cox_censor <- coxph(Surv(OS, 1 - os.status) ~ trt + btmb + pdl1, data = oak)
#
get_marginal_effect(
  trt = "trt", cox_event = cox_event, cox_censor = cox_censor, SE = TRUE,
 M = 1000, n.boot = 10, data = oak, seed = 1, cpp = FALSE, control = clmqControl(n_jobs = 100)
)
#

## End(Not run)
```

`get_point_estimate` *Calculate the marginal treatment effects (only the point estimate) for hazard ratio (HR) adjusting covariates in clinical trials*

Description

This function only estimates the marginal logHR. For a complete estimation including the standard error, see get_marginal_effect.

Usage

```
get_point_estimate(
  trt,
  cox_event,
  cox_censor,
  data,
  M = 1000,
  seed = NULL,
  cpp = TRUE,
  control = clmqControl(),
  verbose = TRUE
)
```

Arguments

`trt`	Character. Variable name of the treatment assignment. Only support two arm trial at the moment.
`cox_event`	Object. A coxph model using the survival time and survival status.
`cox_censor`	Object. A coxph model using the survival time and 1-survival status.
`data`	A data frame used for cox_event and cox_censor.
`M`	Numeric. The number of simulated counterfactual patients. Suggest to set a large M to get robust estimation but this will be time comsuming.
`seed`	Numeric. Random seed for simulation.
`cpp`	Bool. True for using C++ optimization. False for not using C++ optimization. This requires cpp package installed.
`control`	Named list. A list containing control parameters, including memory of remote workers, whether to use nested parallel computation or local multiprocess, number of remote workers/jobs, etc. See details of clmqControl.
`verbose`	Bool. Print status messages. Default: TRUE

Details

Use the simulation approach from Daniel et al. (2020) to estimate the marginal HR when adjusting covariates. This function uses the parallel computation via remote LSF and local multiprocess. Additional feature also includes the C++ optimization that can speed up the calculation.

Value

The marginal beta (logHR)

References

Daniel R, Zhang J, Farewell D. Making apples from oranges: Comparing noncollapsible effect estimators and their standard errors after adjustment for different covariate sets. Biom J. 2021;63(3):528-557. doi:10.1002/bimj.201900297

Examples

```
library(survival)
data("oak")

cox_event <- coxph(Surv(OS, os.status) ~ trt + btmb + pdl1, data = oak)
#
cox_censor <- coxph(Surv(OS, 1 - os.status) ~ trt + btmb + pdl1, data = oak)
#
get_point_estimate(
  trt = "trt", cox_event = cox_event, cox_censor, M = 1000, data = oak,
  seed = 1, cpp = FALSE, control = clmqControl(clmq_hr=FALSE)
)
```

---

get_rmst_estimate *Calculate the marginal restricted mean survival time (RMST) when adjusting covariates in clinical trials*

Description

Estimate the marginal RMST (point estimate) using the Karrison et al.(2018). Standard errors (SE) were estimated using the methods from Zucker (1998), Chen and Tsiatis (2001), and Wei et al.(2023). We implemented both nonparametric(bootstrap) and parametric methods(delta) for SE.

Usage

```
get_rmst_estimate(
  time,
  status,
  trt,
  covariates = NULL,
  tau,
  SE = "delta",
  n.boot = 1000,
  seed = 1
)
```

Arguments

time	A vector containing the event time of the sample.
status	A vector containing the survival status of the sample.
trt	A vector indicating the treatment assignment. 1 for treatment group. 0 for placebo group.
covariates	A data frame containing the covariates. If covariates is NULL, unadjusted RMST is returned.
tau	Numeric. A value for the restricted time or the pre-specified cutoff time point.
SE	Character. If SE = 'boot', SE was estimated using nonparametric bootstrap. If 'delta', SE was estimated using the delta method. Default is 'delta'.
n.boot	Numeric. Number of bootstrap used. Only used if SE = 'boot'.
seed	Numeric. Random seed for bootstrap. Default:1.

Details

Restricted mean survival time is a measure of average survival time up to a specified time point. We adopted the methods from Karrison et al.(2018) for estimating the marginal RMST when adjusting covariates. For the SE, both nonparametric bootstrap and delta method are good for estimation. For the delta estimation of variance,we used a combined estimation including Zucker (1998) and Chen and Tsiatis (2001). SE (delta) = variance from Zucker (1998) + additional variance component from Chen and Tsiatis (2001).The additional variance is coming from treating covariates as random.

Value

A list including marginal RMST and SE.

References

- Karrison T, Kocherginsky M. Restricted mean survival time: Does covariate adjustment improve precision in randomized clinical trials? Clinical Trials. 2018;15(2):178-188. doi:10.1177/1740774518759281
- Zucker, D. M. (1998). Restricted Mean Life with Covariates: Modification and Extension of a Useful Survival Analysis Method. Journal of the American Statistical Association, 93(442), 702–709. https://doi.org/10.1080/01621459.1998.10473722
- Wei, J., Xu, J., Bornkamp, B., Lin, R., Tian, H., Xi, D., ... Roychoudhury, S. (2024). Conditional and Unconditional Treatment Effects in Randomized Clinical Trials: Estimands, Estimation, and Interpretation. Statistics in Biopharmaceutical Research, 16(3), 371–381. https://doi.org/10.1080/19466315.2023.2292774
- Chen, P. and Tsiatis, A. (2001), "Causal Inference on the Difference of the Restricted Mean Lifetime Between Two Groups," Biometrics; 57: 1030–1038. DOI: 10.1111/j.0006-341x.2001.01030.x.

Examples

```
data("oak")
tau <- 26
time <- oak$OS
status <- oak$os.status
```

```
trt <- oak$trt
covariates <- oak[, c("btmb", "pdl1")]
get_rmst_estimate(time, status, trt, covariates, tau, SE = "delta")
```

`get_variance_estimation` *Calculate the variance (SE) of the marginal treatment effects (hazard ratio) adjusting covariates in clinical trials*

Description

Estimate the standard error or variance of the marginal treatment effects using nonparametric bootstrap. Currently, this only supports clustermq for parallel computation.

Usage

```
get_variance_estimation(
  cox_event,
  cox_censor,
  trt,
  data,
  M,
  n.boot,
  seed = NULL,
  cpp = TRUE,
  control = clmqControl(),
  verbose = TRUE
)
```

Arguments

`cox_event`	Object. A coxph model using the survival time and survival status.
`cox_censor`	Object. A coxph model using the survival time and 1-survival status.
`trt`	Character. Variable name of the treatment assignment. Only support two arm trial at the moment.
`data`	A data frame used for cox_event and cox_censor.
`M`	Numeric. The number of simulated counterfactual patients. Suggest to set above 1,000,000 to get robust estimation but it is time comsuming,
`n.boot`	Numeric. Number of bootstrap.
`seed`	Numeric. Random seed for simulation.
`cpp`	Bool. True for using C++ optimization. False for not using C++ optimization. This requires cpp package installed.
`control`	Named list. A list containing control parameters, including memory of remote workers, whether to use nested parallel computation or local multiprocess, number of remote workers/jobs, etc. See details of clmqControl.
`verbose`	Bool. Print status messages. Default: TRUE

Details

If clustermq is not available, we suggest building your own bootstrap like boot and doParallel by using the function – get_point_estimate. This can also get you the SE or variance estimates. If you only run this function, you need to have cox_censor and cox_event in the environment.

Value

A vector containing SE and 95% CI.

References

Daniel R, Zhang J, Farewell D. Making apples from oranges: Comparing noncollapsible effect estimators and their standard errors after adjustment for different covariate sets. Biom J. 2021;63(3):528-557. doi:10.1002/bimj.201900297

Examples

```
## Not run:
#Don't run as it requires LSF scheduler
library(survival)
data("oak")

cox_event <- coxph(Surv(OS, os.status) ~ trt + btmb + pdl1, data = oak)
#
cox_censor <- coxph(Surv(OS, 1 - os.status) ~ trt + btmb + pdl1, data = oak)
#
get_variance_estimation(cox_event, cox_censor,
  trt = "trt", data = oak,
  M = 1000, n.boot = 10, control = clmqControl(), cpp = FALSE
)

## End(Not run)
```

---

oak *Oak trial data*

Description

A dataset from the oak trial used for examples

Usage

```
oak
```

Format

A data frame with 578 rows and 5 variables:

trt Treatment assignment, 1 or 0

btmb Blood-based tumor mutational burden

pdl1 Anti–programmed death ligand 1 (PD-L1) expressed on >=1% tumor cells or tumor-infiltrating immune cells

OS Overall survival time

os.status Overall survival status, 1 is death; 0 is censor

Source

Gandara, D.R., Paul, S.M., Kowanetz, M. et al. Blood-based tumor mutational burden as a predictor of clinical benefit in non-small-cell lung cancer patients treated with atezolizumab. Nat Med 24, 1441–1448 (2018). https://doi.org/10.1038/s41591-018-0134-3

`rmst_delta` *Calculate the variance of the marginal restricted mean survival time (RMST) when adjusting covariates using the delta method*

Description

Standard errors (SE) were estimated using the delta methods from Zucker (1998), Chen and Tsiatis (2001), and Wei et al.(2023).

Usage

```
rmst_delta(fit, time, trt, covariates, tau, surv0, surv1, cumhaz0, cumhaz1)
```

Arguments

`fit`	A coxph object with strata(trt) in the model. See example.
`time`	A vector containing the event time of the sample.
`trt`	A vector indicating the treatment assignment. 1 for treatment group. 0 for placebo group.
`covariates`	A data frame containing the covariates.
`tau`	Numeric. A value for the restricted time or the pre-specified cutoff time point.
`surv0`	A vector containing the cumulative survival function for the placebo group or trt0.
`surv1`	A vector containing the cumulative survival function for the treatment group or trt1.
`cumhaz0`	A data frame containing the cumulative hazard function for the placebo group or trt0.
`cumhaz1`	A data frame containing the cumulative hazard function for the placebo group or trt1.

Details

Restricted mean survival time is a measure of average survival time up to a specified time point. We adopted the methods from Karrison et al.(2018) for estimating the marginal RMST when adjusting covariates. For the SE, both nonparametric bootstrap and delta method are good for estimation. We used the delta estimation from Zucker (1998) but we also included an additional variance component which treats the covariates as random as described in Chen and Tsiatis (2001).

Value

A value of the SE.

References

- Karrison T, Kocherginsky M. Restricted mean survival time: Does covariate adjustment improve precision in randomized clinical trials? Clinical Trials. 2018;15(2):178-188. doi:10.1177/1740774518759281
- Zucker, D. M. (1998). Restricted Mean Life with Covariates: Modification and Extension of a Useful Survival Analysis Method. Journal of the American Statistical Association, 93(442), 702–709. https://doi.org/10.1080/01621459.1998.10473722
- Wei, J., Xu, J., Bornkamp, B., Lin, R., Tian, H., Xi, D., … Roychoudhury, S. (2024). Conditional and Unconditional Treatment Effects in Randomized Clinical Trials: Estimands, Estimation, and Interpretation. Statistics in Biopharmaceutical Research, 16(3), 371–381. https://doi.org/10.1080/19466315.2023.2292774
- Chen, P. and Tsiatis, A. (2001), "Causal Inference on the Difference of the Restricted Mean Lifetime Between Two Groups," Biometrics; 57: 1030–1038. DOI: 10.1111/j.0006-341x.2001.01030.x.

Examples

```
library(survival)
data("oak")

tau <- 26
time <- oak$OS
status <- oak$os.status
trt <- oak$trt
covariates <- oak[, c("btmb", "pdl1")]

dt <- as.data.frame(cbind(time, status, trt, covariates))
fit <- coxph(Surv(time, status) ~ btmb + pdl1 + strata(trt), data = dt)
delta <- rmst_point_estimate(fit, dt = dt, tau)
rmst_delta(fit, time, trt, covariates, tau,
  surv0 = delta$surv0, surv1 = delta$surv1,
  cumhaz0 = delta$cumhaz0, cumhaz1 = delta$cumhaz1
)
```

`rmst_point_estimate` *Calculate the point estimate of the marginal restricted mean survival time (RMST) when adjusting covariates in clinical trials*

Description

Estimate the marginal RMST (point estimate) using the Karrison et al.(2018).

Usage

```
rmst_point_estimate(fit, dt, tau)
```

Arguments

`fit`	A coxph object with strata(trt) in the model. See example.
`dt`	A data frame used for the fit - coxph model including survival time, OS status, trt, and covariates.
`tau`	Numeric. A value for the restricted time or the pre-specified cutoff time point.

Details

Restricted mean survival time is a measure of average survival time up to a specified time point. We adopted the methods from Karrison et al.(2018) for estimating the marginal RMST when adjusting covariates.

Value

A list containing the RMST, cumulative survival function, and cumulative hazard function.

output Marginal RMST

surv0 Cumulative survival function for the placebo group

cumhaz0 Cumulative hazard function for the placebo group

surv1 Cumulative survival function for the treatment group

cumhaz1 Cumulative hazard function for the treatment group

References

- Karrison T, Kocherginsky M. Restricted mean survival time: Does covariate adjustment improve precision in randomized clinical trials? Clinical Trials. 2018;15(2):178-188. doi:10.1177/1740774518759281
- Zucker, D. M. (1998). Restricted Mean Life with Covariates: Modification and Extension of a Useful Survival Analysis Method. Journal of the American Statistical Association, 93(442), 702–709. https://doi.org/10.1080/01621459.1998.10473722
- Wei, J., Xu, J., Bornkamp, B., Lin, R., Tian, H., Xi, D., … Roychoudhury, S. (2024). Conditional and Unconditional Treatment Effects in Randomized Clinical Trials: Estimands, Estimation, and Interpretation. Statistics in Biopharmaceutical Research, 16(3), 371–381. https://doi.org/10.1080/19466315.2023.2292774

- Chen, P. and Tsiatis, A. (2001), "Causal Inference on the Difference of the Restricted Mean Lifetime Between Two Groups," Biometrics; 57: 1030–1038. DOI: 10.1111/j.0006-341x.2001.01030.x.

Examples

```
library(survival)
data("oak")

tau <- 26
time <- oak$OS
status <- oak$os.status
trt <- oak$trt
covariates <- oak[, c("btmb", "pdl1")]
dt <- as.data.frame(cbind(time, status, trt, covariates))
fit <- coxph(Surv(time, status) ~ btmb + pdl1 + strata(trt), data = dt)
delta <- rmst_point_estimate(fit, dt = dt, tau)
delta$output
```

---

rmst_unadjust *Calculate the unadjusted restricted mean survival time (RMST)*

Description

Estimate the unadjusted RMST (point estimate).

Usage

```
rmst_unadjust(time, status, trt, tau)
```

Arguments

time	A vector containing the event time of the sample.
status	A vector containing the survival status of the sample.
trt	A vector indicating the treatment assignment. 1 for treatment group. 0 for placebo group.
tau	Numeric. A value for the restricted time or the pre-specified cutoff time point.

Value

A data frame including the survival time for each trt and the difference. SE were also calculated.

mu0 Mean survival time for trt0

se0 SE of mu0

mu1 Mean survival time for trt1

se1 SE of mu1

delta Difference between mu0 and mu1

se_d SE of delta

Examples

```
data("oak")
tau <- 26
time <- oak$OS
status <- oak$os.status
trt <- oak$trt
covariates <- oak[, c("btmb", "pdl1")]
results <- rmst_unadjust(time, status, trt, tau)
```

`simulate_counterfactuals` *Calculate the potential outcomes using marginal survival causual curves*

Description

Using the marginal survival causal curves from calculate_statistics to simulate the potential outcomes.

Usage

```
simulate_counterfactuals(bh, surv_cond, M, cpp, loadcpp = TRUE)
```

Arguments

`bh`	A data frame from the basehaz function. This is the baseline hazard for the coxph model.
`surv_cond`	A vector containing the marginal causal survival curves from calculate_statistics. Each number is the probability of the surviving the time window (t1,t2],... conditional on surviving the prior corresponding window.
`M`	Numeric. The number of simulated counterfactual patients. Suggest to set above 1,000,000 to get robust estimation but it is time comsuming,
`cpp`	Bool. True for using C++ optimization. False for not using C++ optimization. This requires cpp package installed.
`loadcpp`	Bool. True for loading C++ optimization functions. Default is TRUE. This is only used when cpp = TRUE.

Details

The potential outcomes were simulated by using a Bernoulli distribution from rbinom() and marginal survival causal curves. If M is quite large, we suggest to use C++ optimization to speed up the calculation.

Value

A list containing the simulated event time and simulated status indicator.

References

Daniel R, Zhang J, Farewell D. Making apples from oranges: Comparing noncollapsible effect estimators and their standard errors after adjustment for different covariate sets. Biom J. 2021;63(3):528-557. doi:10.1002/bimj.201900297

Examples

```
library(survival)
data("oak")

cox_event <- coxph(Surv(OS, os.status) ~ trt + btmb + pdl1, data = oak)
#
cox_censor <- coxph(Surv(OS, 1 - os.status) ~ trt + btmb + pdl1, data = oak)

bh <- basehaz(cox_event, centered = FALSE)
s_condi <- calculate_statistics(model = cox_event, trt = "trt")
sim_out_1d <- simulate_counterfactuals(
bh = bh, surv_cond = s_condi$surv_cond0, cpp = FALSE, M = 1000)
```

Index