Package: survdt (via r-universe)

June 5, 2026

Title Improved Methods for Survival Analysis under Double Truncation

Version 0.9.0

Description Contains existing and novel methods for nonparametric analysis and Cox regression analysis with doubly truncated survival data, as described in Vazquez and Xie (2025) <doi:10.1007/s10985-025-09650-5>. Includes survival curves and hazard estimates through nonparametric maximum likelihood estimation, various tests for detecting group differences or non-ignorable sampling bias, and inverse probability weighted Cox regression with several options of nonparametric weights. Also implements diagnostics for key modeling assumptions such as quasi-independent truncation and the positivity assumption. Closed-form standard errors are available for all estimates, i.e. bootstrapping is not required.

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.3.2

Imports ggplot2, igraph, stats, survival

Depends R (>= 2.10)

LazyData true

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Author Omar Vazquez [aut, cre, cph] (ORCID: <https://orcid.org/0009-0009-4137-3333>), Sharon X. Xie [ths] (ORCID: <https://orcid.org/0000-0002-7886-4103>)

Maintainer Omar Vazquez <omar.vazquez@pennmedicine.upenn.edu>

Repository https://cran.r-universe.dev

Date/Publication 2026-05-06 20:50:27 UTC

RemoteUrl https://github.com/cran/survdt

RemoteRef HEAD

RemoteSha a06e20567d9728c6da7ea7af4584d40325e54778

Contents

`aids` *AIDS incubation data*

Description

A doubly truncated dataset of AIDS incubation times from patients with transfusion-acquired HIV collected by the Centers for Disease Control and Prevention (CDC). The sample contains patients who were diagnosed between the discovery of the disease (1982) and the end of study (July 1986), with infection times determined retrospectively from a blood transfusion registry.

Usage

```
aids
```

Format

A data frame with 295 rows and 5 columns:

`incu` The incubation time, i.e. months from HIV infection to AIDS diagnosis.

`age` The age at HIV infection in years.

`ltrunc` The time from HIV infection to the start of data collection (1982), in months. Note that `ltrunc` is negative for patients infected before 1982.

`rtrunc` The time from HIV infection to the end of data collection (July 1986), in months.

`age_gp` A factor with three `age` groups: `<=4`, `4-59`, and `>59` (the baseline level).

Source

Wang, M. -C. (1989). A semiparametric model for randomly truncated data. *Journal of the American Statistical Association*, **84**(407), 742–748.

coxph_indtrunc *Cox regression under quasi-independent double truncation.*

Description

Computes an inverse probability weighted Cox regression estimator using nonparametric weights (see `npmle()`). Includes options for robust estimators with time-varying weights, which introduce an event time stabilization weight (see below).

Usage

```
coxph_indtrunc(
  formula,
  data = NULL,
  strata_formula = NULL,
  ipw_type = c("W-asurv", "W-surv", "W-1", "W-a"),
  se = TRUE,
  robust_se = TRUE,
  basehaz = FALSE,
  restr_times = NULL,
  npmle_fit = NULL,
  robust_infl = NULL,
  simple_labels = FALSE,
  keep_x = FALSE
)

## S3 method for class 'coxph_indtrunc'
vcov(object, ...)

## S3 method for class 'coxph_indtrunc'
coef(object, ...)

## S3 method for class 'coxph_indtrunc'
confint(object, parm, level = 0.95, ...)

## S3 method for class 'coxph_indtrunc'
residuals(
  object,
  type = c("martingale", "deviance"),
  per_subject = type == "deviance",
  ...
)

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

Arguments

formula	A model formula with a Survdt or Survdt2 response. Subjects with any missing values are discarded. See Survdt() and Survdt2().
data	An optional data frame used to evaluate formula and strata_formula.
strata_formula	An optional one-sided formula with stratification variables on the right hand side, to fit a stratified Cox model.
ipw_type	Type of inverse probability weights. 1. "W-asurv" (the default): stabilized robust time-varying weights based on the NPMLE survival function. 2. "W-surv": robust time-varying weights based on the NPMLE survival function. 3. "W-1": standard weights with no time-varying component (Rennert and Xie, 2018) 4. "W-a": stabilized weights (Mandel et al., 2018) 5. For custom time-varying weights, provide a function that takes a numerical vector of times as input and returns a numerical vector of positive weights.
se	Whether to calculate standard errors (default TRUE).
robust_se	Whether to calculate standard errors that are robust to model misspecification (default TRUE). Ignored if se=FALSE.
basehaz	Whether to return the baseline hazard estimate (default FALSE).
restr_times	An optional numerical vector of length two giving a time window to restrict the score function. Values of NA are replaced with the minimum and maximum observed event time respectively, i.e. no restriction. The default setting is no restriction.
npmle_fit	An optional pre-computed npmle object fit on the complete cases. If it was fit with ainv=FALSE (or infl=FALSE when se=TRUE) then it is ignored and a new NPMLE is fit. See npmle().
robust_infl	A square matrix of influence values for the event weight, with one column per individual and one row per follow up time, both in the order that the subject appears in the response variable. Only used if ipw_type is a function and robust_se=TRUE.
simple_labels	Whether to drop the factor names from the stratum labels (Default FALSE).
keep_x	Whether to include the model matrix in the cox_fit component of the return value (Default FALSE).
object, x	A coxph_indtrunc object.
...	Additional arguments passed to other methods.
parm	A numeric or character vector specifying a set of model parameters. Defaults to all identifiable (non-missing) coefficients.
level	Confidence level for confidence intervals.
type	The type of residuals to return.
per_subject	Whether to return per-subject residuals as opposed to per-observation residuals, for models with Survdt2 data. Defaults to TRUE for deviance residuals and FALSE otherwise.

Details

Given a sample of doubly truncated data with event times T_i, truncation times L_i and R_i, and potentially time-varying covariates $X_i(t)$, let $N_i(t) = 1(T_i \leq t)$ be the usual counting process and $Y_i(t) = 1(T_i \geq t)$ be the at-risk process.

`coxph_indtrunc()` fits a Cox model by maximizing an inverse probability weighted partial likelihood, where the double truncation bias is accounted for through nonparametric estimates of $a(t)^{-1} = P(L < T \leq R)/P(L < T \leq R|T = t)$ (see `npmle()`). Assuming `strata_formula=NULL` for sake of illustration, setting `ipw_type="W-1"` specifies non-time-varying weights, so the model is fitted by finding the root of the weighted score function

$$\sum_i \int \frac{1}{\widehat{a}(T_i)}[X_i(t) - \overline{X}_{\widehat{a}}(t,\beta)]dN_i(t)$$

$$\text{where } \overline{X}_{\widehat{a}}(t,\beta) = \frac{\sum_j X_j(t)\widehat{a}(T_j)^{-1}Y_j(t)\exp(X_j(t)^T\beta)}{\sum_j \widehat{a}(T_j)^{-1}Y_j(t)\exp(X_j(t)^T\beta)}.$$

More generally, `coxph_indtrunc()` allows for time-varying weights with a stabilization component $\widehat{w}(t)$ that determines the relative importance of each event time. This can improve robustness and numerical convergence without introducing any bias, in contrast to weight truncation, for example. The weighted score function becomes

$$\sum_i \int \frac{\widehat{w}(t)}{\widehat{a}(T_i)}[X_i(t) - \overline{X}_{\widehat{a}}(t,\beta)]dN_i(t).$$

Several choices of time-varying weights are available:

- `ipw_type="W-surv"`: $\widehat{w}(t) = \widehat{S}(t-)$, where $\widehat{S}(t)$ is the marginal survival function estimated by `npmle()`. Can increase robustness to outliers.
- `ipw_type="W-a"`: $\widehat{w}(t) = \widehat{a}(t)$. Can reduce the influence of extreme inverse probability weights.
- `ipw_type="W-asurv"`: $\widehat{w}(t) = \widehat{a}(t)\widehat{S}(t-)$. Combines the benefits of both weights.
- Custom stabilization weights $\widehat{w}(t)$ can be specified by supplying a function as the input to `ipw_type`. If `robust_se=TRUE`, this also requires the influence values for $\widehat{w}(t)$ to be supplied in `robust_infl`, in order to account for its variability in the standard errors.

Value

`coxph_indtrunc()` returns an object of class `coxph_indtrunc`, i.e. a list with the following components:

`n`	The complete case sample size.
`n_miss`	The number of individuals removed due to missing data.
`n_event`	The number of observed events in the score function.
`coef`	A four-column matrix containing the estimated coefficients, standard errors, z-statistics, and p-values. Coefficient values of `NA` indicate collinearity.
`var_matrix`	The estimated variance matrix for the coefficient estimates. Values of `NA` indicate collinearity.

robust_se	The input value for robust_se.
offset	The offsets from the model formula for complete case observations.
ipw	The estimated inverse probability of non-truncation for each complete case observation.
time_wt	The time-varying component of the inverse probability weights for each complete case observation.
cc_indx	A logical index vector for the complete cases in the original data.
id_indx	A vector of ids matching each observation with a subject among the complete cases.
sort_indx	A numerical vector to sort the complete cases into the order they are used in cox_fit (see below).
s	The stratum value for each complete case observation if a stratified Cox model was fit. Otherwise NULL.
martingale_resid	The weighted martingale residuals for the complete cases, with one entry per observation.
deviance_resid	The weighted deviance residuals for the complete cases, with one entry per subject.
basehaz	For input basehaz=TRUE, a list with first element s containing the stratum values and one additional element for each stratum, in the order of s. Each contains a matrix with following columns: time, cumhaz with the cumulative baseline hazard value (baseline at x=0), var_logcumhaz with the estimated variance of the log(cumhaz), and additional columns, labeled by predictor names, with the estimated covariance between the log(cumhaz) and the regression coefficients. For input basehaz=FALSE, NULL.
timerange	A vector with the minimum and maximum times used in the score function.
eventrange	A vector with the unrestricted range of observed event times.
wt_name	A string giving the name of the weights used in ipw_type.
cox_fit	The coxph object used to obtain the estimates. Note that the data was sorted by strata and time before fitting the model, and the offset may include an additional term of -log(time_wt).
strata_summary	Some summary statistics for each stratum, computed prior to applying restr_times.
call_	The call to coxph_indtrunc().

coxph_indtrunc objects have methods for vcov to extract the variance matrix of estimated coefficients, coef to extract the estimated coefficients, confint to compute confidence intervals for coefficients, residuals for model diagnostics, and print to display a summary of the fitted model.

residuals returns per-subject residuals in the order that the subject first appeared in the original data. The deviance residuals are computed as $\text{sign}(\widehat{M}_{wi})\sqrt{-2[\widehat{M}_{wi} + \delta_{wi}\log(1 - \widehat{M}_{wi}/\delta_{wi})]}$ where $\widehat{M}_{wi}$ is the weighted martingale residual and δ_{wi} is the weighted event indicator, following Therneau et al. (1990).

Stratified Cox model

A stratified Cox model is specified through `strata_formula`. The formula should look like `~svar1 + svar2...` where `svar1` and `svar2` are factor variables defining the strata. This allows the baseline hazard to differ across strata.

Restricted time window in score function

The user can choose to restrict the time range of the score function during estimation. This may be useful when extreme event times have very large weights. Setting the input argument `restr_times=c(t1,t2)` will restrict the range of the integrals in the score function to the interval $[$`t1`, `t2`$]$ instead of the entire event time range. Notably, this time restriction will not introduce any bias in the fitted model.

References

Mandel, M., de Uña-Álvarez, J., Simon, D. K., & Betensky, R. A. (2018). Inverse probability weighted Cox regression for doubly truncated data. *Biometrics*, **74**(2), 481–487.

Rennert, L., & Xie, S. X. (2018). Cox regression model with doubly truncated data. *Biometrics*, **74**(2), 725–733.

Therneau, T. M., Grambsch, P. M., & Fleming, T. R. (1990). Martingale-based residuals for survival models. *Biometrika*, **77**(1), 147–160.

Vazquez, O., & Xie, S. X. (2025). Robust inverse probability weighted estimators for doubly truncated Cox regression with closed-form standard errors. *Lifetime Data Analysis*, **31**(2), 364—393.

See Also

- `plot_coxsurv()`: compute/plot estimates and pointwise confidence intervals for the conditional survival function or cumulative hazard from a fitted Cox model.
- `positivity_sens_indtrunc()`: sensitivity analysis for positivity violations in the Cox model.
- `test_quasiindep_covariates()`: test for dependence between truncation times and the event time or covariates. The nonparametric weights used by `coxph_indtrunc()` requires quasi-independence.
- `vignette("cox-regression-quasiindep", package = "survdt")`: an illustration of Cox regression and related diagnostics applied to the aids data, if vignettes were built when installing survdt.

Examples

```
coxph_indtrunc(Survdt(incu, ltrunc, rtrunc) ~ age_gp,
               data = aids)
```

nonprop_sample — *Simulated data with non-proportional hazards and independent double truncation*

Description

The event times were simulated from a Cox model with hazard function

$$\lambda(t|x, group) = \exp\{0.7group \times 1(t \leq 0.6) - 0.5x\}\lambda_0(t - 0.1)$$

where $\lambda_0(t)$ is the hazard function from a beta(2,2) distribution. The covariates *group* and x were independent bernoulli(0.5) and uniform(0,1) random variates, respectively.

The left truncation time had a uniform(0,0.6) distribution, while the time between the left and right truncation time was an independent uniform(0.1,0.7) random variate. All times were rounded to two decimal places.

Usage

```
nonprop_sample
```

Format

A data frame with 490 rows and 5 columns:

`ltrunc` The left truncation time.

`rtrunc` The right truncation time.

`event_time` The event time.

`group` A binary covariate with non-proportional hazards.

`x` A continuous covariate with proportional hazards.

npmle — *Compute the nonparametric MLE under double truncation*

Description

Computes the nonparametric maximum likelihood estimator (NPMLE) of the pre-truncation event time and truncation time distributions using an EM algorithm.

Usage

```
npmle(
  y,
  data = NULL,
  se = TRUE,
  cumhaz = FALSE,
  ainv = TRUE,
  infl = FALSE,
  max_iter = 1000,
  error = 1e-09,
  init = NULL,
  convergence_warning = TRUE,
  check_identifiability = TRUE
)

is.npmle(x)

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

Arguments

y	A Survdt object or expression. Observations with missing values are discarded. See Survdt().
data	An optional data frame used to evaluate y.
se	Whether to compute standard errors for each estimate (default TRUE).
cumhaz	Whether to also compute the NPMLE of the event time cumulative hazard (default FALSE).
ainv	Whether to return the NPMLE of the normalized inverse selection probabilities at the observed event times (default TRUE).
infl	Whether to return the influence values for each estimate (default FALSE).
max_iter	The maximum number of iterations for the EM algorithm.
error	Tolerance for determining numerical convergence, defined as the relative change in the log likelihood being less than error or the absolute change being less than sqrt(error) at any iteration.
init	Optional event time cdf point masses to initialize the EM algorithm.
convergence_warning	Whether to issue a warning when numerical convergence is not achieved by max_iter iterations (default TRUE).
check_identifiability	Whether to check and return an error if the NPMLE is not identifiable (default TRUE, see below).
x	Any R object.
...	Additional arguments passed to or from other methods.

Details

Given a doubly truncated sample (T_1, L_1, R_1), (T_2, L_2, R_2), ..., (T_n, L_n, R_n), the NPMLE of the (pre-truncation) event time cdf is a right-continuous step function

$$\widehat{F}(t) = \sum_{i=1}^{n} 1(T_i \leq t)\widehat{f}_i$$

with point masses at the observed event times. Assuming (quasi-)independent truncation, these point masses maximize the likelihood function

$$L(f_1, \ldots, f_n) = \prod_{i=1}^{n} \frac{f_i}{\sum_j 1(L_i < T_j \leq R_i) f_j}.$$

Efron and Petrosian (1999) provided a simple EM algorithm to compute the NPMLE.

The point masses of the NPMLE can be expressed as $\widehat{f}_i = 1/[\widehat{a}(T_i)n]$, where $a(t) = P(L < T \leq R|T = t)/P(L < T \leq R)$ is the normalized probability of observing time t within a truncation interval. These inverse selection probabilities are used to correct for double truncation bias in `coxph_indtrunc()`.

Once $\widehat{F}(t)$ is obtained, other quantities such as the survival function, cumulative hazard, and bivariate truncation time cdf point masses are straightforward to compute.

Value

`npmle()` returns an object of class `npmle`, i.e. a list with the following components:

`n`	The complete case sample size.
`n_miss`	The number of discarded observations (with missing values).
`cdf`	A matrix containing the estimated event time cdf evaluated at ordered unique event times, with columns `time`, `cdf`, and `se`.
`cumhaz`	A matrix containing the estimated event time cumulative hazard evaluated at ordered unique event times, with columns `time`, `cumhaz`, and `se`. Set to `NULL` if `cumhaz=FALSE`.
`ainv`	A matrix containing the estimated normalized inverse selection probabilities for each event time (not sorted and including duplicates), with columns `time`, `ainv`, and `se`. Set to `NULL` if `ainv=FALSE`.
`infl_cdf`	A matrix containing the estimated influence function for the cdf, with `[i,j]`th element corresponding to the `i`th ordered unique event time and `j`th unsorted individual. Set to `NULL` if `infl=FALSE`.
`infl_cumhaz`	A matrix containing the estimated influence function for the cumulative hazard, with `[i,j]`th element corresponding to the `i`th ordered unique event time and `j`th unsorted individual. Set to `NULL` if `cumhaz=FALSE` or `infl=FALSE`.
`infl_ainv`	A matrix containing the estimated influence function for the normalized inverse selection probabilities, with `[i,j]`th element corresponding to the `i`th event time (including ties) and `j`th individual, both unsorted. Set to `NULL` if `ainv=FALSE` or `infl=FALSE`.

`masses` A data frame containing the point masses for the event time (`f`) and truncation time (`g`) cdfs and the estimated selection probabilities for each event time (`pt`) and truncation interval (`plr`), with one row per observation. Not sorted.

`converged` Whether the EM algorithm converged before `max_iter` was reached.

`n_iter` The total number of EM algorithm iterations that ran.

`overall_sampling_prob` A named vector with the estimated overall probability of non-truncation in `prob` and its standard error in `se`.

`is.npmle()` returns `TRUE` if `x` inherits from the class `npmle` and `FALSE` otherwise. The `print` method for `npmle` objects displays some sample information and the first few steps of the estimated cdf.

Identifiability

Xiao and Hudgens (2019) provided a necessary and sufficient condition for the likelihood function $L(f_1, \ldots, f_n)$ to have a unique maximizer. By default, this condition is checked by `npmle()` before estimation.

Standard errors

By default, `npmle()` computes variance estimates for the NPMLE using the closed-form standard errors derived by Vazquez and Xie (2025). This is generally more computationally efficient than the resampling methods used by other packages. If fit with `infl=TRUE`, the `npmle` object may be supplied as an input to other survdt functions, which will then not need to re-compute the influence values.

References

Efron, B., & Petrosian, V. (1999). Nonparametric methods for doubly truncated data. *Journal of the American Statistical Association*, **94**(447), 824–834.

de Uña-Álvarez, J., & Van Keilegom, I. (2021). Efron–Petrosian integrals for doubly truncated data with covariates: An asymptotic analysis. *Bernoulli*, **27**(1), 249–273.

Vazquez, O., & Xie, S. X. (2025). Robust inverse probability weighted estimators for doubly truncated Cox regression with closed-form standard errors. *Lifetime Data Analysis*, **31**(2), 364—393.

Xiao, J., & Hudgens, M. G. (2019). On nonparametric maximum likelihood estimation with double truncation. *Biometrika*, **106**(4), 989–996.

See Also

- `plot.npmle()`: compute/plot estimates and pointwise confidence intervals for the event time survival function and cumulative hazard, normalized inverse selection probability weights, and selection probability as a function of time.
- `test_ignorability_indtrunc()`: test for ignorable sampling bias using the NPMLE.
- `test_samplediff_indtrunc()`: test for differences in the event time distributions across groups using the NPMLE.

- `test_quasiindep_ctau()`: test for violations of the quasi-independent truncation assumption.
- `vignette("nonparametric-analysis", package = "survdt")`: an example of various nonparametric analyses applied to the aids data, if vignettes were built when installing survdt.

Examples

```
npmle(Survdt(incu, ltrunc, rtrunc),
      data = aids)
```

`plot.npmle` *Plot estimates from an NPMLE fit*

Description

Given a fitted NPMLE from `npmle()`, plot various quantities related to the event time or truncation time distributions along with pointwise confidence intervals. The estimates and standard errors are computed by `plot.npmle()` if they were not included in the `npmle` fit.

Usage

```
## S3 method for class 'npmle'
plot(
  x,
  ci = TRUE,
  level = 0.95,
  ctype = c("log-log", "log", "plain"),
  target = c("survival", "cumhaz", "ainv", "sel_prob"),
  ...
)
```

Arguments

`x`	An `npmle` object. See `npmle()`.
`ci`	Whether to plot pointwise confidence intervals (default `TRUE`).
`level`	Confidence level for the pointwise confidence intervals (default 0.95).
`ctype`	Type of confidence interval. • `"plain"`: linear confidence intervals from a normal approximation. • `"log"`: back-transformed confidence intervals from the natural log scale. Guaranteed to be non-negative. • `"log-log"` (the default): for the survival function and selection probability, back-transformed confidence intervals from the (natural) log minus log scale, which are guaranteed to be between zero and one. For the normalized inverse probability weights and cumulative hazard, the same confidence intervals as `"log"` are used.

target — Which NPMLE quantity to plot.

- "survival" (the default): the event time survival function.
- "cumhaz": the event time cumulative hazard function.
- "ainv": the normalized inverse probability weights at each observed event time. See npmle().
- "sel_prob": the selection (non-truncation) probability as a function of time.

... — Additional arguments passed to other methods.

Value

After displaying the plot, a list with two elements is returned invisibly. The first element ests is a data frame with the columns time, est, lower, and upper, containing the point estimates and their computed confidence intervals. The second element plot is the plotted object returned by ggplot2::ggplot().

See Also

npmle()

Examples

```
fit <- npmle(Survdt(incu, ltrunc, rtrunc),
             data = aids)
plot(fit)

plot(fit, target = "cumhaz")
```

---

plot_coxsurv *Plot survival or hazard estimates from a Cox model fit*

Description

Given a fitted Cox model from coxph_indtrunc(), plot the conditional survival or cumulative hazard estimates along with pointwise confidence intervals. The baseline hazard and its standard errors are computed by plot_coxsurv() if they were not included in the coxph_indtrunc fit.

Usage

```
plot_coxsurv(
  object,
  newdata,
  ci = TRUE,
  level = 0.95,
  ctype = c("log-log", "log", "plain"),
  target = c("survival", "cumhaz")
)
```

Arguments

object	A coxph_indtrunc object. See coxph_indtrunc().
newdata	A data frame with the covariate values for the plot. The original model formula is applied to extract predictors. Rows that are duplicated or have missing values are discarded.
ci	Whether to plot pointwise confidence intervals (default TRUE). Will re-fit the model if basehaz or its standard errors were not computed.
level	Confidence level for the pointwise confidence intervals (default 0.95).
ctype	Type of confidence interval. • "plain": linear confidence from a normal approximation. • "log": back-transformed confidence intervals from the natural log scale. Guaranteed to be non-negative. • "log-log" (the default): for the survival function, back-transformed confidence intervals from the (natural) log minus log scale, which are guaranteed to be between zero and one. For the cumulative hazard, the same confidence intervals as "log" are used.
target	Which quantity to plot. • "survival" (the default): the conditional survival function. • "cumhaz": the conditional cumulative hazard.

Details

In order to construct the linear predictors at the requested covariate values, the design matrix obtained from evaluating the original model formula on newdata should have the same column names as the design matrix for the fitted model. Issues can occur for categorical variables that were converted to factors by coxph_indtrunc() for estimation. If an error occurs, check that each categorical variable in newdata is a factor variable that is not missing any levels used to fit the model.

Value

After displaying the plot, a list with two elements is returned invisibly. The first element ests is a data frame with the columns time, est, lower, and upper, containing the point estimates and their computed confidence intervals. The second element plot is the plotted object returned by ggplot2::ggplot(). Lines are colored based on the covariate value.

See Also

coxph_indtrunc()

Examples

```
fit <- coxph_indtrunc(Survdt(incu, ltrunc, rtrunc) ~ age_gp,
                      data = aids,
                      basehaz = TRUE)
plot_coxsurv(fit, newdata = aids[c(1,40), ])

plot_coxsurv(fit, newdata = aids[c(1,40), ],
             target = "cumhaz")
```

`positivity_sens_indtrunc` *Sensitivity analysis for positivity violations in a Cox model with quasi-independent double truncation.*

Description

Computes inverse probability weighted sensitivity analysis estimators for the Cox model using non-parametric weights (see `npmle()`). Given a grid of values for the potential amount of baseline probability mass lost due to zero sampling probability, shows how a positivity violation could influence the results of a fitted Cox model.

Usage

```
positivity_sens_indtrunc(
  formula,
  data = NULL,
  strata_formula = NULL,
  trunc_mass = seq(0, 0.9, by = 0.1),
  ipw_type = c("W-asurv", "W-surv", "W-1", "W-a"),
  se = TRUE,
  level = 0.95,
  restr_times = NULL,
  npmle_fit = NULL,
  robust_infl = NULL,
  max_iter = 100,
  error = 1e-07,
  beta_init = NULL,
  center_covariates = FALSE,
  simple_labels = FALSE
)

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

## S3 method for class 'coxph_pos_indtrunc'
plot(x, parm, level, hazratio = FALSE, dropNA = TRUE, ...)
```

Arguments

`formula`	A model formula with a `Survdt` response. Subjects with missing values are discarded. Time-varying covariates are not supported. See `Survdt()`.
`data`	An optional data frame used to evaluate `formula` and `strata_formula`.
`strata_formula`	An optional one-sided formula with stratification variables on the right hand side, for stratified Cox models.

`trunc_mass` A numerical vector containing the grid of truncated baseline mass values to use in the sensitivity analysis.

`ipw_type` Type of inverse probability weights.

1. `"W-asurv"` (the default): stabilized robust time-varying weights based on the NPMLE survival function.
2. `"W-surv"`: robust time-varying weights based on the NPMLE survival function.
3. `"W-1"`: standard weights with no time-varying component (Rennert and Xie, 2018)
4. `"W-a"`: stabilized weights (Mandel et al., 2018)
5. For custom time-varying weights, provide a function that takes a numerical vector of times as input and returns a numerical vector of positive weights.

`se` Whether to calculate (robust) standard errors (default `TRUE`).

`level` Confidence level for constructing sensitivity intervals (default 0.95).

`restr_times` An optional numerical vector of length two giving a time window to restrict the score function. Values of `NA` are replaced with the minimum and maximum observed event time respectively, i.e. no restriction. The default setting is no restriction.

`npmle_fit` An optional pre-computed `npmle` object fit on the complete cases. If it was fit with `ainv=FALSE` (or `infl=FALSE` when `se=TRUE`) then it is ignored and a new NPMLE is fit. See `npmle()`.

`robust_infl` A square matrix of influence values for the event weight, with one column per individual and one row per follow up time, both in the order that they appear in the data. Only used if `ipw_type` is a function and `se=TRUE`.

`max_iter` Maximum number of iterations before non-convergence is declared at a given truncated mass (default 100).

`error` Tolerance for determining numerical convergence of coefficient estimates (default `1e-7`).

`beta_init` Optional numerical vector with initial values for each coefficient estimate.

`center_covariates` Whether to center the covariates before the sensitivity analysis (default `FALSE`). Covariates with values of 0/1 are not centered.

`simple_labels` Whether to drop the factor names from the stratum labels (Default `FALSE`).

`x` A `coxph_pos_indtrunc` object.

`...` Additional arguments passed to other methods.

`parm` A numeric or character vector specifying a set of model parameters. Defaults to all identifiable (non-missing) coefficients.

`hazratio` Whether to plot the hazard ratios exp(beta) and their sensitivity intervals instead of the regression coefficients beta (default `FALSE`).

`dropNA` Whether to remove truncation masses where the estimator did not converge, resulting in missing values, from the plot (default `TRUE`).

Details

Consider a sample of doubly truncated data with event times T_i, truncation times L_i and R_i, and covariates X_i.

A positivity violation occurs when the sampling probability $P(L < T \leq R|T = t)$ is zero for $t \leq a$ and $t > b$, where the observation bounds a and b lie within the event time support. The truncated baseline probability mass is defined as the amount of probability mass right-truncated by the positivity violation at the baseline covariate level, conditional on not being left-truncated by the positivity violation, i.e. $P(T > b|X = 0, T > a)$. Note that the interpretation of this truncated mass depends on how the baseline covariate value is defined, e.g. whether the covariates are centered.

At each specified truncated mass, `positivity_sens_indtrunc()` fits a Cox model that accounts for the positivity violation through an adjustment to the at-risk process (see Vazquez and Xie, 2025). This modified at-risk process is inserted into the weighted score function from `coxph_indtrunc()`.

The end result of the sensitivity analysis is a series of adjusted coefficient estimators at the grid of truncated masses, which show how sensitive the Cox model estimates are to potential positivity violations, as well as sensitivity intervals giving a range of plausible values for the coefficients at the user-supplied confidence `level` for each truncated mass.

In general, the sensitivity analysis estimators may not converge at some truncated mass values. Usually this is due to large weights, near-zero risk scores $\exp(X_i^T\beta)$, and/or truncated masses too close to one. Some options to consider are:

- Changing the baseline covariate value by shifting the covariates or reordering factor levels.
- Using time-varying weights (see `ipw_type`).
- Restricting the time window in the score function through `restr_times` (see below).
- Altering the range and step size of `trunc_mass`.

Value

`positivity_sens_indtrunc()` returns an object of class `coxph_pos_indtrunc`, i.e. a list with the following components:

`n`	The complete case sample size.
`n_miss`	The number of individuals removed due to missing data.
`n_event`	The number of observed events used in the score function.
`level`	The user-specified confidence level for constructing sensitivity intervals.
`trunc_mass`	The truncated baseline masses supplied in `trunc_mass` (sorted). The baseline covariate values are provided in `col_centerings`.
`coef`	A matrix of coefficient estimates with `j`th column corresponding to `trunc_mass[j]`. Missing values indicate non-convergence or collinearity.
`se`	A matrix of standard errors with `j`th column corresponding to `trunc_mass[j]`.
`lower, upper`	Matrices containing the lower and upper sensitivity interval endpoints, with `j`th column corresponding to `trunc_mass[j]`.
`timerange`	A vector with the minimum and maximum times used in the score function.
`eventrange`	A vector with the unrestricted range of observed event times.

wt_name	A string giving the name of the weights used in ipw_type.
col_centerings	A numerical vector with the values used to center the columns of the design matrix if center_covariates=TRUE. Otherwise, a vector of zeros.
strata_summary	Some summary statistics for each stratum, computed prior to applying restr_times.
call_	The call to positivity_sens_indtrunc().

The print method for coxph_pos_indtrunc objects displays a summary of the sensitivity analysis results, while the plot method returns a visualization of the estimators and their sensitivity intervals across different truncated mass values, generated through ggplot2::ggplot().

Stratified Cox model

A stratified Cox model is specified through strata_formula. The formula should look like ~svar1 + svar2... where svar1 and svar2 are factor variables defining the strata. This allows the baseline hazard to differ across strata.

Restricted time window in score function

The user can choose to restrict the time range of the score function during estimation. This may be useful when extreme event times have very large weights. Setting the input argument restr_times=c(t1,t2) will restrict the range of the integrals in the score function to the interval $[$t1, t2$]$ instead of the entire event time range. Notably, this time restriction will not introduce any bias in the fitted model.

References

Vazquez, O., & Xie, S. X. (2025). Robust inverse probability weighted estimators for doubly truncated Cox regression with closed-form standard errors. *Lifetime Data Analysis*, **31**(2), 364—393.

See Also

- coxph_indtrunc(): fitting a Cox model under quasi-independent truncation.
- vignette("cox-regression-quasiindep", package = "survdt"): an illustration of this sensitivity analysis applied to the aids data, if vignettes were built when installing survdt.

Examples

```
sensfit <- positivity_sens_indtrunc(Survdt(incu, ltrunc, rtrunc) ~ age_gp,
                                    data = aids,
                                    trunc_mass = seq(0, 0.11, by = 0.02),
                                    ipw_type = "W-asurv")
sensfit
plot(sensfit)
```

Survdt *Create a basic survival object for doubly truncated data.*

Description

Creates a survival object to contain the event times, left truncation times, and right truncation times from a doubly truncated sample, with one observation per subject.

Typically used as the input for nonparametric analysis methods or as the response variable in a regression model formula.

Usage

```
Survdt(time, ltrunc, rtrunc)

## S3 method for class 'Survdt'
x[i, j, ...]

## S3 method for class 'Survdt'
is.na(x)

is.Survdt(x)

## S3 method for class 'Survdt'
length(x)

## S3 method for class 'Survdt'
as.data.frame(x, ...)

## S3 method for class 'Survdt'
plot(x, colorvar = NULL, sort = TRUE, ...)
```

Arguments

time	A vector of follow up times.
ltrunc	A vector of left truncation times.
rtrunc	A vector of right truncation times.
x	Any R object.
i, j	Row and columns index vectors. The result is a Survdt object if j is missing, otherwise x is treated as a matrix.
...	Additional arguments passed to other methods.
colorvar	An optional vector used to color the dots in the plot. Will be coerced to factor before using.
sort	Whether to sort the data rows by follow up time (and colorvar, if provided) before plotting (default TRUE).

Details

For a given event time of interest T, a sample is doubly truncated when it only includes subjects with $L_i < T_i \leq R_i$ for some random truncation times L_i and R_i. Analyzing this type of data requires information on the ith subject's event time T_i, left truncation time L_i, and right truncation time R_i.

`Survdt()` provides the basic data structure for organizing such data and checking its validity in the `survdt` package. Attempting to run `Survdt()` with inputs that do not satisfy $L < T \leq R$ will result in an error.

Value

- `Survdt()` returns an object of class `Survdt`, with columns `time`, `ltrunc`, and `rtrunc`. Indexing a `Survdt` object by rows will return another `Survdt` object, otherwise it is treated as a matrix. The length of a `Survdt` object is its sample size, and `is.na()` returns a logical index vector for the non-complete cases. Calling `as.data.frame()` on a `Survdt` object will return a data frame stripped of the `Survdt` class attribute.
- `is.Survdt()` returns `TRUE` if `x` inherits from the class `Survdt` and `FALSE` otherwise.
- The plot method for `Survdt` objects returns a simple visualization from `ggplot2::ggplot()` with x-axis time, y-axis subject, rectangles for the truncation intervals, and dots for the event times. The dots can optionally be colored by a factor variable supplied in `colorvar`. Incomplete cases are dropped from the plot (with a warning).

See Also

- `npmle()`, `test_samplediff_indtrunc()`: nonparametric estimation and analysis for doubly truncated data.
- `coxph_indtrunc()`: inverse probability weighted Cox regression.
- `Survdt2()`: to encode time-varying covariates; also `time_split()` for converting data from `Survdt` structure to `Survdt2`.

Examples

```
y <- with(aids, Survdt(incu, ltrunc, rtrunc))
plot(y)
plot(y, colorvar = aids$age_gp)
npmle(y)
```

Survdt2 *Create a survival object for use with time-varying covariates*

Description

Creates a survival object for doubly truncated data with the follow up time for each subject split across multiple observation windows using (start, stop] encoding.

Usage

```
Survdt2(start, stop, event, ltrunc, rtrunc, id)

survdt2to1(x)

## S3 method for class 'Survdt2'
x[i, j, ...]

## S3 method for class 'Survdt2'
is.na(x)

is.Survdt2(x)

## S3 method for class 'Survdt2'
length(x)

## S3 method for class 'Survdt2'
as.data.frame(x, ...)

## S3 method for class 'Survdt2'
plot(x, ...)
```

Arguments

start	A vector of start times for the observation windows.
stop	A vector of stop times for the observation windows.
event	A vector of 0/1 event indicators (1=event).
ltrunc	A vector of left truncation times.
rtrunc	A vector of right truncation times.
id	A vector of subject ids.
x	Any R object.
i, j	Row and columns index vectors. The result is a Survdt2 object if j is missing, otherwise x is treated as a matrix.
...	Additional arguments passed to other methods.

Details

The ith subject's survival data consists of the left and right truncation times L_i and R_i as well as a series of (start, stop] observation windows $(t_{i0}, t_{i1}]$, $(t_{i1}, t_{i2}]$, ..., $(t_{i,j-1}, t_{ij}]$ for some j. Furthermore, each of these observation windows has an associated event indicator that is equal to one if the event time is equal to the stop time, and zero otherwise. Each subject should have an observation window with an event, and multiple events per subject are not allowed. Attempting to run Survdt2() with inputs that do not follow this specification will result in an error.

This data structure is used for Cox regression with time-varying covariates, since the subject's covariate values can vary across observation windows.

Value

- Survdt2() returns an object of class Survdt2, with columns start, stop, event, ltrunc, rtrunc, and id. Indexing a Survdt2 object by rows will return another Survdt2 object, otherwise it is treated as a matrix. The length of a Survdt2 object is the total number of observations across all subjects, and is.na() returns a logical index vector for the non-complete rows. Calling as.data.frame on a Survdt2 object will return a data frame stripped of the Survdt2 class attribute.
- is.Survdt2() returns TRUE if x inherits from the class Survdt2 and FALSE otherwise.
- survdt2to1() takes a Survdt2 object as input and returns the result from calling Survdt() on collapsed data with one row per id. The subjects in the collapsed data are sorted in the order they first appear in the Survdt2 object. If all event indicators for a subject are zero and/or NA, the event time in the collapsed data is set to NA. Lastly, the id values are preserved within the row names of the Survdt object.
- The plot method for Survdt2 first collapses the data into Survdt format using survdt2to1() and then calls plot.Survdt().

See Also

- coxph_indtrunc(): inverse probability weighted Cox regression with time-varying covariates.
- Survdt(): the more basic survival object with one row per subject.

Examples

```
# first create (start, stop] data frame;
# could then apply time transformations
# to covariates
splitdat <- time_split(aids, cut = c(20, 40),
                       stop_name = "incu",
                       id_name = "id",
                       origin = -1)

y1 <- with(splitdat,
           Survdt2(start, incu, event, ltrunc, rtrunc, id))
plot(y1)

y2 <- with(aids,
           Survdt(incu, ltrunc, rtrunc))
max(abs(unclass(survdt2to1(y1)) - unclass(y2)))


coxph_indtrunc(Survdt2(start, incu, event, ltrunc, rtrunc, id) ~ age_gp,
               data = splitdat)
```

`test_ignorability_indtrunc` *Test for non-ignorable sampling bias*

Description

Implements two possible nonparametric tests for non-ignorable sampling bias based on the NPMLE:

1. check for non-constant selection probabilities across the event time range;
2. check for a difference between the estimated event time cumulative distribution functions adjusted and not adjusted for double truncation.

Typically used to determine if a doubly truncated sample must be analyzed with methods that account for truncation bias.

Usage

```
test_ignorability_indtrunc(
  y,
  data = NULL,
  test_type = c("sel_prob", "cdf"),
  level = 0.95,
  B = 2000,
  restr_times = NULL,
  wt_fn = NULL
)

## S3 method for class 'ignorability_indtrunc_test'
plot(x, level, ...)

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

Arguments

`y`	A `Survdt` object or expression, or an `npmle` object. Observations with missing values are discarded. See `Survdt()` and `npmle()`.
`data`	An optional data frame used to evaluate `y`.
`test_type`	The test statistic to use: `"sel_prob"` (default) or `"cdf"`. See below.
`level`	The confidence level for the test (default 0.95).
`B`	The number of simulations from the estimated null distribution used to compute the p-value (default 2000).
`restr_times`	An optional numerical vector of length two giving a time window to restrict the test statistic. Values of `NA` are replaced with the minimum and maximum observed event time respectively, i.e. no restriction. The default setting is no restriction.

wt_fn	An optional function that takes a numerical vector of times as input and returns a vector of weights to apply in the test statistic. If unspecified, all times have equal weight of 1.
x	Any R object.
...	Additional arguments passed to other methods.

Details

The sampling bias from double truncation is called ignorable when the selection (non-truncation) probability is constant over all possible event times. Since all event times have an equal probability of being included in the sample, the sampling mechanism does not alter their distribution, so standard analysis methods can be applied to the data.

Define the normalized selection probability at any time t by $a(t) = P(L < T \leq R|T = t)/P(L < T \leq R)$. Its estimate $\widehat{a}(t)$ is a left-continuous step function with jumps at the observed truncation times. By default, `test_ignorability_indtrunc()` tests for non-ignorable sampling bias through the test statistic

$$\sup_{\min_i T_i \leq t \leq \max_i T_i} |\widehat{a}(t) - 1|$$

with the p-value computed by simulating from the estimated asymptotic null distribution. Another option is to compare two estimated event time cdfs: the NPMLE $\widehat{F}(t)$ which adjusts for double truncation and the empirical cdf $\widehat{F}^*(t)$ which does not adjust for double truncation. The test statistic would then be

$$\sup_t |\widehat{F}(t) - \widehat{F}^*(t)|.$$

Value

`test_ignorability_indtrunc()` returns an object of class `ignorability_indtrunc_test`, i.e. a list with the following components:

n	The complete case sample size.
n_unique	The number of unique event times among complete cases.
n_miss	The number of discarded observations (with missing values).
teststat	The value of the observed test statistic.
pval	The p-value for the test.
critval	The critical value for the test at the user-specified level.
testresult	Whether the test resulted in rejecting the null hypothesis of ignorable sampling bias.
sims	A vector of the B simulations from the estimated null distribution of the test statistic.
test_type	The matched test_type value.
level	The user-supplied level.
B	The number of simulations specified by the user.
timerange	A vector with the minimum and maximum times used in the test.
wt_name	A string giving the name of the user-supplied wt_fn, if any.

`eventrange` A vector with the unrestricted range of observed event times.

`ests` A data frame of estimates to be used by `plot.ignorability_indtrunc_test()`.

The `print` method for `ignorability_indtrunc_test` objects displays a summary of the test results, while the `plot` method returns a visualization comparing the NPMLE and its reference value under the null hypothesis using a uniform confidence band, generated through `ggplot2::ggplot()`.

Restricted time window and weighted tests

When the NPMLE has high variance at extreme event times, the test may suffer from reduced power. One possible remedy is to restrict the time range of the test to some interval $[\tau_1, \tau_2]$ contained in $[\min_i T_i, \max_i T_i]$. Another option is to down-weight event times with large variance through a weight function $w(t)$. The stabilized test statistic will have the form

$$\sup_{\tau_1 \leq t \leq \tau_2} |w(t)(\widehat{h}(t) - h(t))|$$

where $\widehat{h}(t)$ is the NPMLE without assuming the null hypothesis holds and $h(t)$ is the reference value under the null. See `vignette("nonparametric-analysis", package = "survdt")` for an illustration with the aids data assuming vignettes were built when installing survdt.

No effective truncation

A doubly truncated sample has no effective truncation when $\max_i L_i < \min_i T_i$ and $\max_i T_i \leq \min_i R_i$. The estimated selection probability is equal to one across all event times in this case, and its standard errors may not be reliable. Typically one would conclude that the truncation mechanism is ignorable in these types of samples, so standard methods can be applied for data analysis.

References

de Uña-Álvarez, J. (2023). Testing for an ignorable sampling bias under random double truncation. *Statistics in Medicine*, **42**(20), 3732–3744.

Vazquez, O., & Xie, S. X. (2025). Robust inverse probability weighted estimators for doubly truncated Cox regression with closed-form standard errors. *Lifetime Data Analysis*, **31**(2), 364—393.

See Also

- `npmle()`: the nonparametric maximum likelihood estimator which forms the basis for these tests.
- `vignette("nonparametric-analysis", package = "survdt")`: an illustration of ignorability tests applied to the aids data, if vignettes were built when installing survdt.

Examples

```
igtest <- test_ignorability_indtrunc(Survdt(incu, ltrunc, rtrunc),
                                     data = aids)
igtest
plot(igtest)
```

`test_quasiindep_covariates` *Test for dependence between truncation times and the event time or covariates*

Description

Implements a two-stage test for dependence between the truncation times and the event time and covariates that first tests within covariate strata and then across strata.

Typically used to check the underlying assumptions for inverse probability weighted regression with NPMLE weights, e.g. as used in `coxph_indtrunc()`.

Usage

```
test_quasiindep_covariates(
  formula,
  data = NULL,
  level = 0.95,
  B = 2000,
  restr_times = NULL,
  wt_fn = NULL,
  max_iter = NULL,
  simple_labels = FALSE
)

## S3 method for class 'quasiindep_strat_test'
plot(x, level, ...)

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

Arguments

`formula`	A formula with a `Survdt` response and stratification variables on the right hand side. Note that subtraction will not remove stratification variables here. Subjects with missing values are discarded. See `Survdt()`.
`data`	An optional data frame used to evaluate `formula`.
`level`	The confidence level for the test (default 0.95).
`B`	The number of simulations from the estimated null distribution used to compute the p-value (default 2000).
`restr_times`	An optional numerical vector of length two giving a time window to restrict the test statistic. Values of `NA` are replaced with the minimum and maximum observed event time respectively, i.e. no restriction. The default setting is no restriction.

wt_fn	An optional function that takes a numerical vector of times as input and returns a vector of weights to apply in the test statistic. If unspecified, all times have equal weight of 1.
max_iter	The maximum number of iterations to use within each call to npmle(). If missing then the default setting of npmle() is used.
simple_labels	Whether to drop the factor names from the stratum labels (Default FALSE).
x	Any R object.
...	Additional arguments passed to other methods.

Details

Implements a nonparametric test for dependent truncation to check if the selection probability depends on either the event times or covariates. The test has two stages:

1. Test for dependence between event and truncation times stratified by covariate groups using conditional Kendall's tau (see test_quasiindep_ctau()). The aggregate p-value for this stage is computed from a chi-squared distribution.
2. Test whether the selection probability depends on covariates by comparing the stratified and unstratified selection probabilities estimated using the NPMLE. Let $\widehat{\pi}(t|s)$ be the estimate in stratum s, and $\widehat{\pi}(t)$ be the unstratified estimate. Both are left-continuous step functions with jumps at observed truncation times. By default, test_quasiindep_covariates() uses the test statistic
$$\max_s \sup_{\min_i T_i \le t \le \max_i T_i} |\widehat{\pi}(t|s) - \widehat{\pi}(t)|$$
to compare selection probabilities across strata. The p-value for this stage is computed by simulating from the estimated asymptotic null distribution.

Value

An object of class quasiindep_strat_test, i.e. a list with the following components:

n	The complete case sample size.
n_unique	The number of unique event times among complete cases.
n_miss	The number of discarded observations (with missing values).
teststat	The value of the observed test statistic for the across-strata test (stage 2).
pval	The p-value for the across-strata test.
critval	The critical value for the across-strata test at the user-specified level.
testresult	Whether the across-strata test results in rejecting the null hypothesis of quasi-independence.
sims	A vector of the B simulations from the estimated null distribution of the across-strata test statistic.
level	The user-supplied level.
B	The number of simulations specified by the user.
timerange	A vector with the minimum and maximum times used in the across-strata test.
wt_name	A string giving the name of the user-suppled wt_fn, if any.

`eventrange`	A vector with the unrestricted range of observed event times.
`strata_ests`	A data frame of estimates to be used by `plot.quasiindep_strat_test()`.
`strata_ctau`	A data frame of the within-strata test results (stage 1). Includes stratum-specific tests by conditional Kendall's tau as well as the aggregate p-value for testing within-strata quasi-independence in all strata simultaneously.

The `print` method for `quasiindep_strat_test` objects displays a summary of the test results, while the `plot` method returns a visualization comparing the stratified and unstratified NPMLE from stage 2 using a uniform confidence band, generated through `ggplot2::ggplot()`.

Restricted time window and weighted tests

When the NPMLE has high variance at extreme event times, the test may suffer from reduced power. One possible remedy is to restrict the time range of the test to some interval $[\tau_1, \tau_2]$ contained in $[\min_i T_i, \max_i T_i]$. Another option is to down-weight event times with large variance through a weight function $w(t)$. The stabilized test statistic will have the form

$$\sup_{\tau_1 \leq t \leq \tau_2} |w(t)(\widehat{h}(t) - h(t))|$$

where $\widehat{h}(t)$ is the NPMLE without assuming the null hypothesis holds and $h(t)$ is the reference value under the null. See `vignette("nonparametric-analysis", package = "survdt")` for an illustration with the aids data assuming vignettes were built when installing survdt.

No effective truncation

A doubly truncated sample has no effective truncation when $\max_i L_i < \min_i T_i$ and $\max_i T_i \leq \min_i R_i$. The estimated selection probability is equal to one across all event times in this case, and its standard errors may not be reliable. Typically one would conclude that the truncation mechanism is ignorable in these types of samples, so standard methods can be applied for data analysis.

References

Vazquez, O., & Xie, S. X. (2025). Robust inverse probability weighted estimators for doubly truncated Cox regression with closed-form standard errors. *Lifetime Data Analysis*, **31**(2), 364—393.

See Also

- `test_quasiindep_ctau()`: the conditional Kendall's tau test for quasi-dependence without covariates, which is the basis for stage 1.
- `npmle()`: the nonparametric maximum likelihood estimator which is the basis for stage 2.
- `vignette("cox-regression-quasiindep", package = "survdt")`: an illustration of this quasi-independence test applied to the aids data, if vignettes were built when installing survdt.

Examples

```
indtest <- test_quasiindep_covariates(Survdt(incu, ltrunc, rtrunc) ~ age_gp,
                                      data = aids)
indtest
plot(indtest)
```

`test_quasiindep_ctau` *Test for dependence between truncation times and the event time*

Description

Computes the conditional Kendall's tau rank correlation between the event time and each of the two truncation times, accounting for double truncation. Then tests whether both correlations are equal to zero.

Typically used to determine if nonparametric analysis can be done using the NPMLE, which requires quasi-independence (see `npmle()`).

Usage

```
test_quasiindep_ctau(y, data = NULL, singular_thresh = 1e-09)

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

Arguments

`y`	A `Survdt` object or expression. Observations with missing values are discarded. See `Survdt()`.
`data`	An optional data frame used to evaluate `y`.
`singular_thresh`	The numerical threshold for determining whether the variance matrix for the correlations is singular (default `1e-9`). This can occur when the truncation times are a constant distance from each other.
`x`	Any R object.
`...`	Additional arguments passed to other methods.

Details

The conditional Kendall's tau rank correlation between the left truncation time and event time is estimated by the average of

$$\text{sign}\{(T_i - T_j)(L_i - L_j)\}$$

across all comparable pairs of subjects (i, j). A pair is called comparable if $L_j < T_i \leq R_j$ and $L_i < T_j \leq R_i$, which ensures that the order of the pair's times is not influenced by truncation under quasi-independence. Similarly, the conditional Kendall's tau rank correlation between the right truncation time and event time is estimated by the average of

$$\text{sign}\{(T_i - T_j)(R_i - R_j)\}$$

across the same comparable pairs of subjects (i, j).

Under quasi-independent truncation, both conditional Kendall's tau rank correlations should converge to zero in large samples.

Value

`test_quasiindep_ctau()` returns an object of class `quasiindep_ctau_test`, i.e. a list with the following components:

`tau`	A numerical vector of length two containing the estimated conditional Kendall's tau rank correlation (1) between the event time and left truncation time, and (2) between the event time and right truncation time.
`teststat`	The chi-squared test statistic for testing the null hypothesis that both correlations are zero.
`df`	The degrees of freedom for `teststat`.
`pval`	The p-value for testing that both correlations are zero.
`var_matrix`	The estimated variance matrix for the two conditional Kendall's tau correlation estimates.
`singular`	Whether `var_matrix` was determined to be singular.
`prob_comp`	The fraction of subject pairs that are comparable.

The `print` method for `quasiindep_ctau_test` objects displays a summary of the test results.

References

Martin, E. C., & Betensky, R. A. (2005). Testing quasi-independence of failure and truncation times via conditional Kendall's tau. *Journal of the American Statistical Association*, **100**(470), 484–492.

See Also

- `test_quasiindep_covariates()`: a test for dependent truncation when covariates are involved, e.g. when aiming to analyze data with `coxph_indtrunc()`.

Examples

```
test_quasiindep_ctau(Survdt(incu, ltrunc, rtrunc), aids)
```

---

test_samplediff_indtrunc

Test for survival differences across groups

Description

Tests for any difference in event time distribution by comparing the stratified and unstratified estimates obtained using the NPMLE.

Usage

```
test_samplediff_indtrunc(
  formula,
  data = NULL,
  test_type = c("survival", "cumhaz"),
  level = 0.95,
  B = 2000,
  restr_times = NULL,
  wt_fn = NULL,
  max_iter = NULL,
  simple_labels = FALSE
)

## S3 method for class 'samplediff_indtrunc_test'
plot(x, level, ...)

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

Arguments

formula	A formula with a Survdt response and stratification variables on the right hand side. Note that subtraction will not remove stratification variables here. Subjects with missing values are discarded. See Survdt().
data	An optional data frame used to evaluate formula.
test_type	The test statistic to use: "survival" (default) or "cumhaz". See below.
level	The confidence level for the test (default 0.95).
B	The number of simulations from the estimated null distribution used to compute the p-value (default 2000).
restr_times	An optional numerical vector of length two giving a time window to restrict the test statistic. Values of NA are replaced with the minimum and maximum observed event time respectively, i.e. no restriction. The default setting is no restriction.
wt_fn	An optional function that takes a numerical vector of times as input and returns a vector of weights to apply in the test statistic. If unspecified, all times have equal weight of 1.
max_iter	The maximum number of iterations to use within each call to npmle(). If missing then the default setting of npmle() is used.
simple_labels	Whether to drop the factor names from the stratum labels (Default FALSE).
x	Any R object.
...	Additional arguments passed to other methods.

Details

Let $\widehat{S}(t|s)$ be the NPMLE for the survival function in stratum s. By default, `test_samplediff_indtrunc()` tests for event time distribution differences through the test statistic

$$\max_s \sup_{\min_i T_i \le t \le \max_i T_i} |\widehat{S}(t|s) - \widehat{S}(t)|$$

where $\widehat{S}(t)$ is the unstratified NPMLE. Another option is to compare the stratified cumulative hazards $\widehat{\Lambda}(t|s)$ with the unstratified NPMLE $\widehat{\Lambda}(t)$ through the test statistic

$$\max_s \sup_{\min_i T_i \le t \le \max_i T_i} |\widehat{\Lambda}(t|s) - \widehat{\Lambda}(t)|.$$

The p-value in all cases is computed by simulating from the estimated asymptotic null distribution.

Value

`test_samplediff_indtrunc()` returns an object of class `samplediff_indtrunc_test`, i.e. a list with the following components:

`n`	The complete case sample size.
`n_unique`	The number of unique event times among complete cases.
`n_miss`	The number of discarded observations (with missing values).
`teststat`	The value of the observed test statistic.
`pval`	The p-value for the test.
`critval`	The critical value for the test at the user-specified level.
`testresult`	Whether the test results in rejecting the null hypothesis of no differences across strata.
`sims`	A vector of the `B` simulations from the estimated null distribution of the test statistic.
`test_type`	The matched `test_type` value.
`level`	The user-supplied `level`.
`B`	The number of simulations specified by the user.
`timerange`	A vector with the minimum and maximum times used in the test.
`wt_name`	A string giving the name of the user-supplied `wt_fn`, if any.
`eventrange`	A vector with the unrestricted range of observed event times.
`strata_ests`	A data frame of estimates to be used by `plot.samplediff_indtrunc_test()`.
`strata_summary`	Some summary statistics for each stratum.

The `print` method for `samplediff_indtrunc_test` objects displays a summary of the test results, while the `plot` method returns a visualization comparing the stratified and unstratified NPMLE using a uniform confidence band, generated through `ggplot2::ggplot()`.

Restricted time window and weighted tests

When the NPMLE has high variance at extreme event times, the test may suffer from reduced power. One possible remedy is to restrict the time range of the test to some interval $[\tau_1, \tau_2]$ contained in $[\min_i T_i, \max_i T_i]$. Another option is to down-weight event times with large variance through a weight function $w(t)$. The stabilized test statistic will have the form

$$\sup_{\tau_1 \leq t \leq \tau_2} |w(t)(\widehat{h}(t) - h(t))|$$

where $\widehat{h}(t)$ is the NPMLE without assuming the null hypothesis holds and $h(t)$ is the reference value under the null. See `vignette("nonparametric-analysis", package = "survdt")` for an illustration with the aids data assuming vignettes were built when installing survdt.

References

Vazquez, O., & Xie, S. X. (2025). Robust inverse probability weighted estimators for doubly truncated Cox regression with closed-form standard errors. *Lifetime Data Analysis*, **31**(2), 364—393.

See Also

- `npmle()`: the nonparametric maximum likelihood estimator which forms the basis for these tests.
- `vignette("nonparametric-analysis", package = "survdt")`: an illustration of group tests applied to the aids data, if vignettes were built when installing survdt.

Examples

```
difftest <- test_samplediff_indtrunc(Survdt(incu, ltrunc, rtrunc) ~ age_gp,
                                     data = aids,
                                     restr_times = c(NA, 70))
difftest
plot(difftest)
```

`time_split` *Split rows in a data frame into multiple observations at specified time intervals*

Description

Splits each record in the data frame into a series of observation windows based on the supplied cut times. Can be used to reshape data with time-varying covariates to the appropriate format for `Survdt2()`. The main workhorse is a modified call to `survival::survSplit()`.

Usage

```
time_split(
  data,
  cut,
  stop_name,
  start_name = "start",
  event_name = "event",
  id_name = NULL,
  origin = 0
)
```

Arguments

data	A data frame to be split.
cut	A numeric vector of cut times.
stop_name	The name of the existing event time or stop time column in data.
start_name	The name for the start time column to be created in the split data. If it matches a column in data (meaning the data was previously split) then event_name must also match a column in data.
event_name	The name for the event indicator column to be created in the split data. See start_name for special requirements.
id_name	An optional name to create an id column in the split data. If this column already exists in data its values will be used as ids. If the data is already in (start, stop] format, do not use this argument.
origin	The time that the follow up starts at. Defaults to zero.

Details

A basic use case of time_split() is to take a data frame with one row per subject and transform it into a data frame with multiple rows per subject, each corresponding to a specific observation time window. The event time information observed in each window is encoded by four potentially new variables:

- start: the starting time for the observation window.
- stop: the end time for the observation window.
- event: equal to one when the subject's event occurred at stop, and zero otherwise.
- id: the label that identifies which subject the data corresponds to.

In this use case, the name of the event time column in data should always be provided, as well as the name for the new id column if one does not already exist. After cut is sorted, the ith subject's observation windows will be (origin, cut[1]], (cut[1], cut[2]], ..., (cut[j], T_i] for some j$\leq$length(cut).

Another use case of time_split() is to introduce additional splits in a previously split data frame. The names for the existing start, stop, and event columns should be provided to time_split() in this case. Requesting a new id variable for previously cut data will return an error, since the new ids would not respect the prior groupings.

Value

A split data frame with (start, stop] records. The new column names are taken from the input arguments of `time_split()`.

See Also

- `Survdt2()`: create survival objects with split data.
- `coxph_indtrunc()`: Cox regression with time-varying covariates.
- `survdt2to1()`: collapse data back to one row per subject.

Examples

```
splitdat1 <- time_split(aids, cut = c(20, 40),
                        stop_name = "incu",
                        id_name = "id",
                        origin = -1)
tail(splitdat1)

splitdat2 <- time_split(splitdat1, cut = c(10, 30),
                        start_name = "start",
                        stop_name = "incu",
                        event_name = "event",
                        id_name = "id",
                        origin = -1)
tail(splitdat2)
```

Index