

Package: ertte (via r-universe)

October 5, 2026

Title Exposure-Response Tools for Time-to-Event Models

Version 0.1.1

Description Provides estimation tools for exposure-response models of time-to-event endpoints based on the 'survival' package: parametric accelerated failure time model fitting and prediction with AIC-based distribution selection, stepwise covariate modelling, and simulation-based visual predictive checks. Tested and supported for the exponential, Weibull, log-normal, and log-logistic distributions. For a model-agnostic mini-language to visualise exposure-response models (including those fitted with 'ertte'), see the companion package 'erplots'.

License MIT + file LICENSE

Language en-GB

Encoding UTF-8

Roxygen list(markdown = TRUE)

Imports dplyr, mvtnorm, rlang, stats, survival, tibble, withr

Suggests erplots, knitr, rmarkdown, testthat (>= 3.0.0)

URL <https://github.com/djnavarro/ertte>, <https://ertte.djnavarro.net/>

BugReports <https://github.com/djnavarro/ertte/issues>

Depends R (>= 4.1.0)

LazyData true

Config/testthat/edition 3

Config/Needs/website djnavarro/waeponwifestre, rmarkdown

Config/roxygen2/version 8.1.0

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

Date/Publication 2026-10-04 23:37:10 UTC

RemoteUrl <https://github.com/djnavarro/ertte>

RemoteRef HEAD

RemoteSha 0cb6d7b2c630db3d391ebaa9bbbc02156e83d03b

Contents

ertte_aft	2
ertte_aft_select_distribution	3
ertte_coxph	4
ertte_data	5
ertte_fun	6
ertte_landmark	8
ertte_power	9
ertte_predict	10
ertte_rmst	12
ertte_scm	14
ertte_term	16
simulate.ertte_model	17
Index	19

ertte_aft	<i>AFT regression modelling</i>
-----------	---------------------------------

Description

Fits a parametric accelerated failure time (AFT) regression model for time-to-event data.

Usage

```
ertte_aft(formula, data, dist = "weibull", ...)
```

Arguments

formula	Model formula specifying the regression model, e.g. <code>Surv(time, event) ~ exposure</code> .
data	Data frame containing the variables of interest.
dist	The AFT distribution type to use (default is "weibull").
...	Other arguments passed to <code>survival::survreg()</code> .

Details

The `ertte_aft()` function is a thin wrapper around `survival::survreg()`, used to build parametric AFT models for time-to-event data. Unlike the original, it caches the input data frame within the returned model object, ensuring that the data set remains accessible to downstream tools that have access to the model object but not necessarily the original data frame.

Like the `survreg()` function upon which it is based, `ertte_aft()` fits a log-location-scale AFT model,

$$\log(t) = \mu + \sigma w$$

where μ is the linear predictor, σ is the scale parameter, and w follows a fixed base distribution determined by `dist`:

- "weibull"/"exponential": w follows a standard extreme-value distribution.
- "lognormal": w follows a standard normal distribution.
- "loglogistic": w follows a standard logistic distribution.

In principle `ertte_aft()` could support the full range of distributions available to `survreg()`, but it is important to note that the extended functionality provided by the `ertte` package, and hooks into other exposure-response tools supplied by other packages such as `erplots`, have not been thoroughly tested beyond four core cases. Tested values for the `dist` argument are `weibull`, `exponential`, `lognormal`, and `loglogistic`. It is for this reason that the output object from `ertte_aft()` contains additional classes: the `ertte`-specific tools are not necessarily valid for all `survreg` objects, only those for which it has been explicitly defined.

Nevertheless, because the return value is genuinely a `survreg` object, albeit with some additional information and metadata stored internally, all the usual methods for survival regression models work unchanged, without needing any `ertte`-specific equivalent. This includes `summary()`, `coef()`, `vcov()`, `confint()`, `predict()`, `AIC()`, `BIC()`, `logLik()`, and `anova()`. Additional methods supplied via the `ertte`-specific classes include `ertte_predict()`, `ertte_fun()`, and `simulate()`.

Value

A `survreg` object with additional `ertte_aft` and `ertte_model` classes used to supply additional methods.

Examples

```
# fit a Weibull AFT model; because the ertte package reexports Surv(),
# there is no need to load the survival package to build this model
mod <- ertte_aft(Surv(time, event) ~ aucss, ertte_data)
mod

# other AFT distributions are also supported
mod_ln <- ertte_aft(Surv(time, event) ~ aucss, ertte_data, dist = "lognormal")
mod_ln
```

```
ertte_aft_select_distribution
```

Select an AFT distribution by AIC

Description

Fits each candidate AFT distribution to the same formula/data and selects the best fit by AIC.

Usage

```
ertte_aft_select_distribution(
  formula,
  data,
  candidates = c("exponential", "weibull", "lognormal", "loglogistic")
)
```

Arguments

formula	Model formula, as for <code>ertte_aft()</code>
data	Data set
candidates	Character vector of candidate dist values to try. Defaults to all four tested/supported distributions.

Details

Ties (to floating point) are broken by the order candidates is given in, i.e. the first-listed of the tied candidates is returned as model.

candidates must be a non-empty character vector with no missing values; each element must also separately name one of the tested/supported distributions (see `ertte_aft()`'s dist argument). candidates = character(0) errors rather than silently fitting nothing and returning a degenerate list(comparison = <0-row tibble>, model = NULL).

Value

A list with two elements: comparison (a tibble with one row per candidate: dist, logLik, aic, bic, converged, sorted by AIC) and model (the best-fitting ertte_aft model, i.e. the one with lowest AIC).

Examples

```
# (uses ertte_aft() internally for each candidate distribution)
cmp <- ertte_aft_select_distribution(Surv(time, event) ~ aucss, ertte_data)
cmp$comparison
cmp$model
```

ertte_coxph

Cox proportional hazard modelling

Description

Fits a semi-parametric Cox proportional-hazards regression of time-to-event data.

Usage

```
ertte_coxph(formula, data, ...)
```

Arguments

formula	Model formula specifying the regression model, e.g. <code>Surv(time, event) ~ exposure</code> .
data	Data frame containing the variables of interest.
...	Other arguments passed to <code>survival::coxph()</code> .

Details

The `ertte_coxph()` function is a thin wrapper around `survival::coxph()`, used to build semi-parametric proportional hazard models for time-to-event data. One slight difference is that the call to `coxph()` always sets `model = TRUE`, because downstream methods supplied by the `ertte` package require the model frame to be accessible from within the return object. Along similar lines, `ertte_coxph()` caches the input data frame within the returned model object, ensuring that it remains accessible to downstream tools that have access to the model object but not necessarily the original data frame.

Because the return value is a `coxph` object, all the usual methods for survival regression models work unchanged, without needing an `ertte`-specific equivalent. This includes `summary()`, `coef()`, `vcov()`, `confint()`, `predict()`, `AIC()`, `BIC()`, `logLik()`, and `anova()`. Additional methods supplied via the `ertte`-specific classes include `ertte_predict()`, `ertte_fun()`, and `simulate()`.

Value

A `coxph` object with with additional `ertte_coxph` and `ertte_model` classes used to supply additional methods.

Examples

```
mod <- ertte_coxph(Surv(time, event) ~ aucss, ertte_data)
mod
```

ertte_data

Sample simulated data for exposure-response time-to-event models

Description

Sample simulated data for exposure-response time-to-event models

Usage

```
ertte_data
```

Format

A data frame with columns:

id Identifier

sex Sex

age Age

weight Weight

dose Nominal dose, units not specified

treatment Treatment

aucss AUCss
cmaxss Cmax,ss
time Time to event or right-censoring, in days
event Event indicator (1 = event observed, 0 = right-censored)
admin_censor Administrative censoring cutoff (days) – fixed at 180 for every subject, regardless of whether that subject had an event. Demonstrates the `sensor_time` argument to `simulate.ertte_model()`, since (unlike `time`) it's knowable for event rows too.

Details

This simulated dataset is entirely synthetic. `time/event` were generated from a Weibull AFT model with an exposure (`aucss`) and sex effect, administrative censoring at 180 days (retained as `admin_censor`), and a little independent random (dropout) censoring (not retained, since – like most real TTE data – a subject's dropout time isn't observable once an event has occurred). You can find the data generating code in the package source code.

Examples

```
ertte_data
```

```
ertte_fun
```

Prediction function for an exposure-response TTE model

Description

Returns a function that evaluates a fitted `ertte` model's survival function at user-specified data, times, and (optionally) counterfactual parameters, without needing to refit the model.

Usage

```
ertte_fun(object, ...)

## S3 method for class 'ertte_aft'
ertte_fun(object, ...)

## S3 method for class 'ertte_coxph'
ertte_fun(object, ...)
```

Arguments

<code>object</code>	An <code>ertte</code> model.
<code>...</code>	Passed to methods.

Details

`ertte_fun()` is a generic function, with methods for each of the supported exposure-response time-to-event model classes.

Value

A function with arguments `data`, `time`, and `param`:

- The `data` argument should be a data frame or tibble; defaults to `object$data` (the data the model was fitted to) if not supplied.
- The `time` argument gives the time(s) at which to evaluate the survival function; recycled against `data`.
- The `param` argument should be a vector of location coefficients; defaults to `coef(object)` (the fitted coefficients) if not supplied.

The function returns a vector of survival probabilities.

AFT models

The `ertte_aft` method takes a fitted AFT model as input and returns a function that evaluates the survival function $S(t)$ at user-specified parameters, data, and times. Note that the `scale` parameter is always taken from the fitted object, and not passed via `param`.

Cox PH models

The `ertte_coxph` method takes a fitted Cox model as input and similarly returns a function that evaluates the survival probabilities. More precisely, it returns

$$S(t|x) = S_0(t)^{\exp((x-\bar{x})'\beta)}$$

where x is the vector of covariates, $\bar{x}$ is the mean of the covariates in the fitted model, β is the vector of coefficients, and $(x - \bar{x})'\beta$ (a matrix multiplication) is the linear predictor that $S_0(t)$ is raised to the power of.

The baseline survival function $S_0(t)$ refers to the fitted baseline survival curve, calculated by calling `survival::basehaz()` with `centered = TRUE` and held constant beyond the last observed time, consistent with the approach adopted by `ertte_predict()` for Cox proportional hazard models.

Analogous to the AFT model case, `param` is only used to vary the linear predictor: the baseline hazard is always taken from the fitted model, and not recomputed for a hypothetical `param` value as that would require refitting the model.

Examples

```
# fit an AFT model
mod_aft <- ertte_aft(Surv(time, event) ~ aucss, ertte_data)
mod_aft_fun <- ertte_fun(mod_aft)

# when called with no arguments it reproduces the fitted
# model's own survival predictions
s1 <- mod_aft_fun(time = 60)
s1[1:5]

# when called with an explicit parameter vector, the model
# is evaluated at those modified parameters
par_new <- coef(mod_aft)
```

```

par_new["(Intercept)"] <- par_new["(Intercept)"] + 1
s2 <- mod_aft_fun(param = par_new, time = 60)
s2[1:5]

# the same logic applies for Cox regression models
mod_cox <- ertte_coxph(Surv(time, event) ~ aucss, ertte_data)
mod_cox_fun <- ertte_fun(mod_cox)
s3 <- mod_cox_fun(time = 60)
s3[1:5]

```

ertte_landmark	<i>Landmark event-probability predictions for exposure-response TTE models</i>
----------------	--

Description

Reduces the survival curve for an exposure-response time-to-event model to a binary landmark event probability at a single fixed time.

Usage

```
ertte_landmark(object, newdata = NULL, landmark_time, conf_level = 0.95, ...)
```

Arguments

object	An ertte model.
newdata	Data frame containing cases to be predicted. Defaults to the data the model was fitted to.
landmark_time	A single positive number.
conf_level	Confidence level for the intervals. Defaults to .95.
...	Passed to ertte_predict() .

Details

In some time-to-event analyses it is convenient to reduce a time-to-event endpoint to a binary landmark response: for every row in the data, the landmark response describes the modelled probability that the event occurs at or before a specified `landmark_time` defined by the analyst. The landmark event probability is simply one minus the survival probability at that time, and is computed using `ertte_predict()`. However, unlike the typical usage of `ertte_predict()` in which a vector of times is passed, `landmark_time` must be a single fixed value – a landmark is by definition evaluated at one time.

Because the event probability is a decreasing transform of the survival probability ($1 - \text{survival}$), the confidence interval is built by swapping `ertte_predict()`'s bounds: its lower survival bound becomes the upper event-probability bound, and vice versa.

Value

A tibble with one row per row of newdata. In addition to storing the original columns from newdata, it contains a column storing the landmark_time itself, a fit_resp column with the estimated probability of observing the event at or before the landmark time, and ci_lower, and ci_upper columns specifying the confidence interval.

Examples

```
mod <- ertte_aft(Surv(time, event) ~ aucss, ertte_data)
ertte_landmark(mod, ertte_data[1:5, ], landmark_time = 180)

mod_cox <- ertte_coxph(Surv(time, event) ~ aucss, ertte_data)
ertte_landmark(mod_cox, ertte_data[1:5, ], landmark_time = 180)
```

ertte_power	<i>Power-function covariate transform for exposure-response TTE models</i>
-------------	--

Description

A formula helper used to enter a continuous covariate as a power transformed covariate rather than a plain linear term.

Usage

```
ertte_power(x, ref = NULL)
```

Arguments

x	A numeric covariate. Every non-missing value must be strictly positive.
ref	A single strictly positive reference value. Defaults to the median value of x.

Details

Both `ertte_aft()` and `ertte_coxph()` are linear in their covariates on the model's natural scale (i.e., log-time for AFT models and log-hazard-ratio for Cox models). A power-function covariate effect is defined as follows for an AFT model:

$$t = t_{\text{ref}} \times (x/x_{\text{ref}})^{\theta}$$

The equivalent for Cox models is as follows:

$$h(t|x) = h_0(t) \times (x/x_{\text{ref}})^{\theta}$$

Once log-transformed, these become linear terms in $\log(x / \text{ref})$.

So `ertte_power(x)` reduces the power-function parameterisation to an ordinary covariate column: the fitted `survreg()/coxph()` coefficient on `ertte_power(x)` is the power exponent θ directly, and its ordinary Wald confidence interval (from `confint()/summary()`) is exactly the confidence interval on θ .

`ertte_power()` requires every non-missing value of x to be strictly positive, so cannot be used for covariates with a placebo/zero-dose group (e.g. `dose`, `aucss`, `cmaxss` in `ertte_data`). It is not generally appropriate for the exposure itself, only for (some) covariate terms in the TTE model.

Nothing prevents combining a plain linear term (`age`) and a power term (`ertte_power(age)`) for the same underlying variable in the same model or SCM candidate set – term handling works on formula term-labels, not variable semantics, so this is left to the user’s judgement.

`ref` is fixed at fitting time from the data `ertte_power()` is evaluated on, and reused (not recomputed) when the fitted model is used to predict or simulate on new data, via an internal `stats::makepredictcall()` method – the same mechanism `stats::poly()` uses for this purpose.

Value

A numeric vector equal to $\log(x / \text{ref})$, classed as an “`ertte_power`” object, with the value of `ref` stored as an attribute.

Examples

```
mod <- ertte_aft(Surv(time, event) ~ aucss + ertte_power(age), ertte_data)
summary(mod)

# reference value used for the power transform
attr(ertte_power(ertte_data$age), "ref")
```

ertte_predict

Survival-probability predictions for exposure-response TTE models

Description

Computes fitted survival probabilities $S(t)$ and confidence intervals from a fitted `ertte` model, for one or more rows of `newdata` at one or more time values.

Usage

```
ertte_predict(object, ...)

## S3 method for class 'ertte_aft'
ertte_predict(object, newdata = NULL, time, conf_level = 0.95, ...)

## S3 method for class 'ertte_coxph'
ertte_predict(object, newdata = NULL, time, conf_level = 0.95, ...)
```

Arguments

<code>object</code>	An ertte model
<code>...</code>	Passed to methods
<code>newdata</code>	Data frame containing cases to be predicted. Defaults to the data the model was fitted to.
<code>time</code>	Numeric vector of times at which to compute survival probabilities
<code>conf_level</code>	Confidence level for the intervals. Defaults to .95. Must be a single number between 0 and 1 (inclusive); other values error.

Details

`ertte_predict()` is a generic function, with two supplied methods, one for parametric AFT exposure-response models (i.e., `ertte_aft` classed objects) and another for Cox proportional hazard exposure-response models (i.e., objects with class `ertte_coxph`).

For AFT models, it computes the linear predictor and standard error using `predict()`, and then converts to a survival probability $S(t)$:

$$S(t) = 1 - F((\log(t) - \mu)/\sigma)$$

where $F(\cdot)$ denotes the cumulative distribution function for the AFT base distribution (e.g., Weibull), μ denotes the linear predictor, and σ denotes the scale parameter. Confidence intervals are constructed using Wald intervals on the linear predictor and back-transformed onto the survival probability scale. Parameter uncertainty in the scale parameter is not propagated.

For Cox models, `survival::survfit()` is used to compute the survival probability. When time exceeds the last observed follow-up time, the survival function is held constant (i.e., step-function extrapolation). Confidence intervals are calculated using the `conf.type = "log"` transform, which specifies Wald intervals on $\log(-\log(S))$, as this is better suited to a probability bounded in $[0, 1]$. Because of this, it should be noted that the confidence intervals computed for a Cox model are not directly comparable to those computed for AFT models.

Value

A tibble with one row per combination of `newdata` row and `time`

Examples

```
# predictions for an AFT model
mod_aft <- ertte_aft(Surv(time, event) ~ aucss, ertte_data)
ertte_predict(mod_aft, ertte_data[1:5, ], time = c(30, 60, 90))

# predictions for a Cox model
mod_cox <- ertte_coxph(Surv(time, event) ~ aucss, ertte_data)
ertte_predict(mod_cox, ertte_data[1:5, ], time = c(30, 60, 90))
```

ertte_rmst	<i>Restricted mean survival time predictions for exposure-response TTE models</i>
------------	---

Description

Computes restricted mean survival time (RMST), the area under a fitted ertte model's survival curve up to one or more fixed horizons τ , with confidence intervals.

Usage

```
ertte_rmst(object, newdata = NULL, tau, conf_level = 0.95, ...)

## S3 method for class 'ertte_aft'
ertte_rmst(object, newdata = NULL, tau, conf_level = 0.95, ...)

## S3 method for class 'ertte_coxph'
ertte_rmst(object, newdata = NULL, tau, conf_level = 0.95, ...)
```

Arguments

object	An ertte model, as returned by ertte_aft() or ertte_coxph() .
newdata	Data frame containing cases to be predicted. Defaults to the data the model was fitted to.
tau	Numeric vector of restriction horizons at which to compute RMST.
conf_level	Confidence level for the intervals. Defaults to .95. Must be a single number between 0 and 1 (inclusive); other values error.
...	Passed to methods.

Details

`ertte_rmst()` is used to reduce a time-to-event endpoint to a scalar exposure-response value, in this case the restricted mean survival time (RMST). The RMST for some horizon time τ is defined as the area under the survival function calculated up to the horizon:

$$\text{RMST}(\tau) = \int_0^{\tau} S(t) dt$$

Because this calculation is performed differently for parametric AFT models and the Cox semiparametric proportional hazard model, `ertte_rmst()` is a generic function with two defined methods, one for each kind of model supported by the ertte package.

For both model types, confidence intervals are symmetric Wald intervals on the RMST scale. This is not automatically bounded to $[\emptyset, \tau]$ the way that the survival probability intervals for `ertte_predict()` are bounded to $[\emptyset, 1]$. An unclipped Wald interval on RMST can, in principle, dip below 0 or exceed τ for small samples or near-boundary cases.

Value

A tibble with one row per combination of newdata row and tau, plus `fit_rmst`, `ci_lower`, and `ci_upper`.

AFT method

The `ertte_aft` method computes `fit_rmst` by numerically integrating the closed-form survival function via `stats::integrate()`. The integration variable is $u = \log(t)$ rather than t itself: adaptive quadrature over the raw time scale, t from 0 to τ , can fail silently for an extreme τ , whereas substituting $t = \exp(u)$ and integrating u from $-\infty$ to $\log(\tau)$ is numerically well-behaved.

The standard error `se_rmst` uses the delta method: it's the absolute gradient of RMST with respect to the linear predictor μ , scaled by the standard error of μ itself,

$$\text{se_rmst} = \left| \frac{d\text{RMST}(\tau)}{d\mu} \right| \times \text{se}(\mu)$$

where the gradient is itself obtained by numerical integration of the survival function's derivative with respect to μ .

`fit_rmst` itself stays numerically reliable well beyond the observed follow-up range, but `se_rmst`/the confidence interval, also computed via numerical integration, can become unreliable for a very extreme horizon – `ertte_rmst()` warns when any value of τ exceeds 10,000x the last observed follow-up time in the fitting data.

Cox PH method

The `ertte_coxph` method delegates the work to `survival::survfit()`. If any value of τ exceeds the last observed follow-up time across the whole fitted cohort, `ertte_rmst()` warns that the assumption that survival stays flat beyond the observed range may be unreliable, since this assumption has a larger effect on an area than on a point-in-time prediction.

Because the fitted baseline hazard (and therefore every covariate-adjusted survival curve) is a right-continuous step function, `fit_rmst` is an *exact* finite sum of rectangle areas between consecutive jump times up to τ : it is not a numerical-quadrature approximation. The standard error column `se_rmst` comes from a delta method that reuses the same rectangle/tail-weighted construction as the Greenwood-based RMST variance method in `survival::survmean()`, but with the variance-increment term replaced by increments of the profile-specific `std.err(t)^2` returned by `survfit()`. In contrast, `survmean()`'s own Greenwood term is based on population-level risk sets shared across every covariate profile, which understates uncertainty for a profile far from the mean covariate values.

Examples

```
# RMST for a parametric AFT model
mod <- ertte_aft(Surv(time, event) ~ aucss, ertte_data)
ertte_rmst(mod, ertte_data[1:5, ], tau = c(60, 90))

# RMST for a semiparametric Cox PH model
mod_cox <- ertte_coxph(Surv(time, event) ~ aucss, ertte_data)
```

```
ertte_rmst(mod_cox, ertte_data[1:5, ], tau = c(60, 90))
```

ertte_scm

Stepwise covariate modelling for exposure-response TTE models

Description

Iteratively adds (`ertte_scm_forward()`) or removes (`ertte_scm_backward()`) covariate terms from an ertte model one at a time, selecting each step's term by a likelihood-ratio p-value, AIC, or BIC, and logging the search in a history object retrievable via `ertte_scm_history()`.

Usage

```
ertte_scm_forward(
  mod,
  candidates,
  threshold = 0.01,
  criterion = "p-value",
  seed = NULL
)

ertte_scm_backward(
  mod,
  candidates,
  threshold = 0.001,
  criterion = "p-value",
  seed = NULL
)

ertte_scm_history(mod)
```

Arguments

<code>mod</code>	An ertte model object
<code>candidates</code>	Character vector with list of candidate terms
<code>threshold</code>	Threshold to test against. Used only when <code>criterion = "p-value"</code> (the default); ignored otherwise. Defaults to 0.01 for <code>ertte_scm_forward()</code> and 0.001 for <code>ertte_scm_backward()</code> .
<code>criterion</code>	Model selection criterion. One of "p-value" (default), "aic", or "bic".
<code>seed</code>	Optional seed to control the order candidate terms are tested within a step. If NULL (the default), one is chosen automatically (see "Candidate test order and seed" below for when this actually matters).

Details

Three model selection criteria are available via the `criterion` argument:

- "p-value" (default): Models are compared with a likelihood-ratio Chi-squared test. A term is added if its likelihood-ratio p-value falls below `threshold` (forward) or removed if its p-value exceeds `threshold` (backward). When multiple candidates satisfy the threshold within a step, the one with the most extreme p-value is chosen.
- "aic": A term is added (forward) or removed (backward) if doing so strictly decreases AIC relative to the current model. When multiple candidates improve AIC, the one yielding the lowest AIC is chosen.
- "bic": Same as "aic", but using BIC as the criterion.

When `criterion` is "aic" or "bic", the `threshold` argument has no effect and is ignored, and `term_p_value` is left NA in the history for every candidate tested that step (the likelihood-ratio test isn't computed, since it plays no role in selection). The `model_aic` and `model_bic` columns are always recorded regardless of which criterion drove selection, and the history's `criterion` column records which one was used for each forward/backward step.

Value

For `ertte_scm_forward()` and `ertte_scm_backward()`, the updated ertte model is returned, with the SCM history log updated internally. For `ertte_scm_history()`, a data frame is returned containing the SCM history log.

Handling problem candidates

A candidate that can't be fit for a given step (e.g. it's aliased with a term already in the model, references a variable missing from the data, or fails to converge) is skipped for that step with a warning, rather than aborting the whole search or being silently selected.

Candidate test order and seed

`seed` exists as a safety measure against run-to-run variation in the order candidate terms are tested within a step (`sample()`, shuffled before testing one at a time). Model fitting itself (`survival::survreg()`) is deterministic given a starting formula, so `seed` only matters in the (essentially measure-zero) case of an exact p-value tie between competing candidates within a step – see the companion `erglm` package's equivalent documentation for the full rationale, which applies unchanged here.

Examples

```
# Forward addition by p-value selection
mod0 <- ertte_aft(Surv(time, event) ~ aucss, ertte_data)
mod1 <- ertte_scm_forward(mod0, candidates = c("sex", "dose"))
ertte_scm_history(mod1)

# Backward elimination by p-value selection
mod2 <- ertte_aft(Surv(time, event) ~ aucss + sex + dose, ertte_data)
mod3 <- ertte_scm_backward(mod2, candidates = c("sex", "dose"))
ertte_scm_history(mod3)
```

```
# AIC-based forward addition/backward elimination instead of p-value
mod4 <- ertte_scm_forward(mod0, candidates = c("sex", "dose"), criterion = "aic")
mod5 <- ertte_scm_backward(mod4, candidates = c("sex", "dose"), criterion = "bic")
ertte_scm_history(mod5)
```

ertte_term	<i>Add or remove a covariate term from an exposure-response TTE model</i>
------------	---

Description

Add or remove a single covariate term from an existing ertte model, returning a new fitted model object.

Usage

```
ertte_add_term(mod, term, quiet = FALSE)

ertte_remove_term(mod, term, quiet = FALSE)
```

Arguments

mod	An ertte model object.
term	A one-sided formula naming the term to add/remove, e.g. <code>~ sex</code> .
quiet	Should the warning issued when the term can't be added/removed (because it's already in the model / isn't in the model, respectively) be suppressed? Defaults to FALSE.

Details

These functions are not typically called directly; they underpin `ertte_scm_forward()` and `ertte_scm_backward()`, used to add or remove a single term from a time-to-event regression model. `mod` is refit using the matching engine constructor based on its class, so these functions work for both `ertte_aft` and `ertte_coxph` models. Regardless of whether the model is a parametric AFT model or a Cox proportional hazard model, the term to be added or removed is defined by a single one-sided formula. Categorical covariates enter the model as factor levels, whereas continuous covariates enter as linear terms by default. However, for a power-function parameterisation (theta such that $T \sim (x / \text{ref})^\theta$ on the AFT time scale, or $h(t|x) \sim h_0(t) * (x / \text{ref})^\theta$ on the Cox hazard scale), the covariate can be wrapped in `ertte_power()`, e.g. `~ ertte_power(age)`.

Value

An ertte model object. If the term can't be added/removed (see `quiet`), the original `mod` is returned unchanged.

Examples

```
mod1 <- ertte_aft(Surv(time, event) ~ aucss, ertte_data)
mod2 <- ertte_add_term(mod1, ~ sex)
mod3 <- ertte_remove_term(mod2, ~ sex)
```

simulate.ertte_model *Simulate from an exposure-response TTE model*

Description

simulate() method for ertte_model objects. A single shared method covers both ertte_aft and ertte_coxph fits – there’s no separate simulate.ertte_coxph() – with the engine-specific simulation mechanics applied automatically based on the fitted object’s class (see Details).

Usage

```
## S3 method for class 'ertte_model'
simulate(
  object,
  nsim = 100,
  seed = NULL,
  newdata = NULL,
  censor_time = NULL,
  ...
)
```

Arguments

object	An ertte model object, as returned by ertte_aft() or ertte_coxph()
nsim	Number of simulation replicates. Defaults to 100.
seed	Optional seed. If NULL (the default), one is chosen automatically and reported via a message (since it determines the actual simulated values returned).
newdata	Data frame to simulate from. Defaults to the data the model was fitted to. Must contain the original response columns (time/event, as named in the model’s Surv() call) – see Details.
censor_time	Optional administrative/maximum-follow-up time(s) to cap simulated event times at, applied uniformly to every row regardless of whether that row observed an event. Either NULL (the default – see Details for the fallback behaviour), a single number (recycled across all rows of newdata), or a numeric vector of length nrow(newdata) giving each row’s own administrative follow-up time.
...	Unused, present for compatibility with the simulate() generic

Details

Coefficients are sampled from the asymptotic sampling distribution implied by vcov(object). Event times are then drawn by inverse-CDF sampling, differing by engine: for ertte_aft fits, directly from the fitted log-location-scale AFT distribution implied by the model’s dist; for ertte_coxph fits, by inverting the fitted baseline cumulative hazard ([survival::basehaz\(\)](#), held fixed regardless of the sampled coefficient draw – the same simplification [ertte_fun.ertte_coxph\(\)](#) makes for a user-supplied param).

The resulting *raw* (uncensored) simulated event time is then censored by `sensor_time` if supplied (`sim_time <- pmin(sim_time_raw, sensor_time)`, with `sim_event` set accordingly) – this is the accurate case, whenever a genuine per-row (or study-wide constant) administrative follow-up time is known, since it caps every row (event or censored) against its true censoring horizon.

Absent a supplied `sensor_time` (the default, `NULL`), row’s own observed event status determines the fallback: rows that were *censored* in `newdata` have their observed exit time used as the cap (`sim_time <- pmin(sim_time_raw, observed_time)`), since that observed exit time genuinely is when censoring happened – an exact match, not an approximation. Rows that had an observed *event*, however, are left **uncensored** in the simulation: their observed exit time is when the event actually happened, not their administrative censoring horizon (which was necessarily later, and typically isn’t recorded once an event has occurred) – capping simulated draws there would leak the observed event day into the simulation and bias a simulated-vs-observed comparison (e.g. a visual predictive check) toward looking more similar than the fitted model actually implies. This remains an approximation for event rows (no censoring is applied at all, absent better information), but avoids that specific bias.

Value

A tibble with one row per observation per replicate: `dat_id/sim_id`, sampled `coef_*` columns, `sim_time` (the simulated event/censoring time), and `sim_event` (1 = event, 0 = censored).

Examples

```
mod <- ertte_aft(Surv(time, event) ~ aucss, ertte_data)
sim <- simulate(mod, nsim = 20, seed = 1234)
sim

mod_cox <- ertte_coxph(Surv(time, event) ~ aucss, ertte_data)
sim_cox <- simulate(mod_cox, nsim = 20, seed = 1234)
sim_cox

# a genuine per-row administrative censoring time -- ertte_data's
# `admin_censor` column records the fixed 180-day study cutoff used
# to generate it, known regardless of whether a subject had an event
sim_admin <- simulate(mod, nsim = 20, seed = 1234, sensor_time = ertte_data$admin_censor)
sim_admin
```

Index

* datasets

- ertte_data, [5](#)
- ertte_add_term(ertte_term), [16](#)
- ertte_aft, [2](#)
- ertte_aft(), [4](#), [9](#), [12](#), [17](#)
- ertte_aft_select_distribution, [3](#)
- ertte_coxph, [4](#)
- ertte_coxph(), [9](#), [12](#), [17](#)
- ertte_data, [5](#)
- ertte_fun, [6](#)
- ertte_fun.ertte_coxph(), [17](#)
- ertte_landmark, [8](#)
- ertte_power, [9](#)
- ertte_power(), [16](#)
- ertte_predict, [10](#)
- ertte_predict(), [8](#)
- ertte_remove_term(ertte_term), [16](#)
- ertte_rmst, [12](#)
- ertte_scm, [14](#)
- ertte_scm_backward(ertte_scm), [14](#)
- ertte_scm_backward(), [16](#)
- ertte_scm_forward(ertte_scm), [14](#)
- ertte_scm_forward(), [16](#)
- ertte_scm_history(ertte_scm), [14](#)
- ertte_term, [16](#)
- simulate.ertte_model, [17](#)
- simulate.ertte_model(), [6](#)
- stats::integrate(), [13](#)
- survival::basehaz(), [17](#)
- survival::coxph(), [4](#), [5](#)
- survival::survreg(), [2](#)