Package: survivalGPU (via r-universe)

November 18, 2026

Title Fast and Scalable Implementations for Survival Models

Version 0.1.0

Description Performs survival analysis on graphics processing units (GPUs) using Cox proportional hazards and weighted cumulative exposure (WCE) models, with support for bootstrap resampling and memory management. The Cox model is described in Cox (1972) <doi:10.1111/j.2517-6161.1972.tb00899.x> and the WCE model in Sylvestre and Abrahamowicz (2009) <doi:10.1002/sim.3701>. Computations are delegated to 'PyTorch' via 'reticulate'; the WCE model additionally requires 'PyKeOps', which is not available natively on Windows, but can be used there through the Windows Subsystem for Linux ('WSL').

License LGPL (>= 2.1)

Language en-US

Encoding UTF-8

Depends R (>= 4.1.0)

Imports graphics, reticulate, stats, survival, utils

Suggests dplyr, knitr, rmarkdown, survminer, testthat (>= 3.0.0), WCE

Config/testthat/edition 3

Config/Needs/website pkgdown

VignetteBuilder knitr

SystemRequirements Python (>= 3.10) with the 'torch', 'numpy', 'pandas', 'scipy', 'matplotlib', 'beartype' and 'jaxtyping' packages; 'pykeops' (optional, required only for wceGPU(); not available natively on Windows but can be used there through the Windows Subsystem for Linux ('WSL'))

URL https://jeanfeydy.github.io/survivalGPU/, https://github.com/jeanfeydy/survivalGPU

BugReports https://github.com/jeanfeydy/survivalGPU/issues

Config/roxygen2/version 8.1.0

NeedsCompilation no

Author Jean FEYDY [aut], Alexis van STRAATEN [aut], Antoine Poirot-Bourdain [aut, cre], Terry THERNEAU [ctb, cph] (Author of the 'survival' package, from which parts of the R code are adapted), Pierre SABATIER [ctb], Anne-Sophie JANNOT [ctb]

Maintainer Antoine Poirot-Bourdain <antoine.poirot.bourdain@gmail.com>

Config/pak/sysreqs python3

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

Date/Publication 2026-11-17 13:10:02 UTC

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

RemoteRef HEAD

RemoteSha a232aed71945f4832969bff794a3805a899d64de

Contents

survivalGPU-package 2
bootstrap 3
coxphGPU 5
HR 8
predict.coxphGPU 9
print.coxphGPU 10
residuals.coxphGPU 11
use_cuda 12
wceGPU 13

Index **17**

`survivalGPU-package` *survivalGPU: Fast Survival Analysis*

Description

Fits Cox proportional hazards and weighted cumulative exposure (WCE) models on graphics processing units (GPUs), with bootstrap confidence intervals computed in batches.

To learn more about survivalGPU, start with `vignette("survivalGPU")`.

Functions

- `use_cuda()`: tells whether computations run on a GPU.
- `coxphGPU()`: fits a Cox proportional hazards model.
- `wceGPU()`: fits a weighted cumulative exposure model.
- `bootstrap()`: adds bootstrap confidence intervals to a fitted model.
- `HR()`: hazard ratio between two exposure histories, for a WCE model.

Author(s)

Maintainer: Antoine Poirot-Bourdain <antoine.poirot.bourdain@gmail.com>

Authors:

- Antoine Poirot-Bourdain <antoine.poirot.bourdain@gmail.com>
- Jean FEYDY <jean.feydy@inria.fr>
- Alexis van STRAATEN <alexis.vanstraaten@outlook.fr>

Other contributors:

- Terry THERNEAU (Author of the 'survival' package, from which parts of the R code are adapted) [contributor, copyright holder]
- Pierre SABATIER [contributor]
- Anne-Sophie JANNOT [contributor]

See Also

Useful links:

- https://jeanfeydy.github.io/survivalGPU/
- https://github.com/jeanfeydy/survivalGPU
- Report bugs at https://github.com/jeanfeydy/survivalGPU/issues

---

`bootstrap` *Bootstrap Confidence Intervals for a Fitted Model*

Description

Adds bootstrap confidence intervals to a model fitted by `coxphGPU()` or `wceGPU()`. The model is refitted on `R` resamples of the patients, in batches on the GPU.

Usage

```
bootstrap(object, ...)

## S3 method for class 'coxphGPU'
bootstrap(
  object,
  R,
  batchsize = 0,
  init = TRUE,
  control,
  device = NULL,
  double_precision = TRUE,
  ...
)
```

```
## S3 method for class 'wceGPU'
bootstrap(
  object,
  R,
  batchsize = 0,
  init = TRUE,
  device = NULL,
  double_precision = TRUE,
  ...
)
```

Arguments

`object`	A `coxphGPU` or `wceGPU` object.
`...`	Not used.
`R`	Number of bootstrap replicates.
`batchsize`	Number of bootstrap replicates processed at a time. `0` (default): all of them at once. If you run out of memory, try `100`, `10` or `1`.
`init`	Starting coefficients of the refits. `TRUE` (default): the coefficients of `object`. `FALSE`: zero. Or a numeric vector. For a `wceGPU` model fitted with several `nknots` candidates, whose spline coefficients have different lengths, `TRUE` starts from zero and a numeric vector is not allowed.
`control`	(`coxphGPU` only) A `survival::coxph.control()` object. Default: the one used to fit `object`.
`device`	Device on which the refits run (`"cpu"` or `"cuda"`). `NULL` (default): the GPU if one is available.
`double_precision`	Logical. `TRUE` (default) for 64-bit floats; `FALSE` for 32-bit floats, faster on most GPUs.

Details

For a `coxphGPU` model, `object` must have been fit with `x = TRUE` and `patient_id` (`coxphGPU(..., x = TRUE, patient_id = "id")`). The bootstrap resamples patients, not rows, so that all the rows of a patient (e.g. with time-varying covariates) are kept together: `patient_id` tells which rows belong to the same patient. The design matrix, the response and the patient of each row are stored in the fit, so the bootstrap always uses exactly the data the model was fitted on, including after `subset =` or rows dropped for missing values.

For a `wceGPU` model, `object` must have been fit with `x = TRUE` (`wceGPU(..., x = TRUE)`): the columns used by the model are stored in the fit, and the bootstrap resamples them, so it always uses exactly the data the model was fitted on. The spline-basis features are rebuilt from them with the same knot configuration; the patient identifier is the `id` column given at the fit. If the model was fitted with several `nknots` candidates, the number of knots is selected again in every bootstrap replicate (see `bootstrap.best.nknots`), so that the confidence intervals account for the uncertainty on that choice.

Value

A copy of `object` with its bootstrap components updated; the point estimates (coefficients, variance, residuals, ...) are unchanged, so `summary()`, `plot()` and `HR()` show the bootstrap confidence intervals.

- For a `coxphGPU` model: `coef_bootstrap` (one row per replicate) and `nbootstraps`.
- For a `wceGPU` model: `bootstrap_beta.hat.covariates`, `WCEmat_bootstrap`, `WCEmat_CI`, `coef_CI`, `bootstrap.best.nknots`, `nbootstraps`, `is_bootstraps`, and `bootstrap_est`/`est_CI` for a single `nknots` value.

Examples

```
## Not run:
library(survival)
drugdata <- WCE::drugdata
fit <- coxphGPU(Surv(Start, Stop, Event) ~ sex + age,
                data = drugdata, x = TRUE, patient_id = "Id")
fit_boot <- bootstrap(fit, R = 1000, batchsize = 200)
summary(fit_boot)

## End(Not run)
## Not run:
drugdata <- WCE::drugdata
fit <- wceGPU(data = drugdata, nknots = 1:3, cutoff = 90, id = "Id",
              event = "Event", start = "Start", stop = "Stop",
              expos = "dose", covariates = c("age", "sex"), x = TRUE)
fit_boot <- bootstrap(fit, R = 500, batchsize = 100)
summary(fit_boot)
table(fit_boot$bootstrap.best.nknots)

## End(Not run)
```

---

coxphGPU *Fast Cox Proportional Hazards Model*

Description

Fits a Cox proportional hazards model on the GPU (or the CPU), with the interface of `survival::coxph()`. Only point estimates are computed: call `bootstrap()` on the result for bootstrap confidence intervals, and inspect it with `summary()`.

Usage

```
coxphGPU(formula, data, ties = c("efron", "breslow"), init, control,
         singular.ok = TRUE, model = FALSE, x = FALSE, y = TRUE,
         patient_id = NULL, ...)

## S3 method for class 'coxphGPU'
summary(object, ..., conf.int = 0.95, scale = 1)
```

Arguments

formula	A formula with a survival object (see survival::Surv()) on the left of ~ and the terms on the right.
data	a data.frame in which to interpret the variables named in the formula, or in the subset and the weights argument.
ties	a character string specifying the method for tie handling. If there are no tied death times all the methods are equivalent. The Efron approximation is used as the default, it is more accurate when dealing with tied death times, and is as efficient computationally. (But see below for multi-state models.) The "exact partial likelihood" is equivalent to a conditional logistic model, and is appropriate when the times are a small set of discrete values.
init	vector of initial values of the iteration. Default initial value is zero for all variables.
control	Object of class coxph.control specifying iteration limit and other control options. Default is coxph.control(...).
singular.ok	logical value indicating how to handle collinearity in the model matrix. If TRUE, the program will automatically skip over columns of the X matrix that are linear combinations of earlier columns. In this case the coefficients for such columns will be NA, and the variance matrix will contain zeros. For ancillary calculations, such as the linear predictor, the missing coefficients are treated as zeros.
model	logical value: if TRUE, the model frame is returned in component model.
x	logical value: if TRUE, the x matrix is returned in component x.
y	logical value: if TRUE, the response vector is returned in component y.
patient_id	Name of the column of data that identifies each patient (subject). Required by bootstrap(), which resamples patients rather than rows, so that all the rows of a patient (e.g. with time-varying covariates) are kept together. With one row per patient, a column of row indices (data$id <- seq_len(nrow(data))) can be used.
...	For coxphGPU(): device ("cpu" or "cuda"; default: the GPU if one is available), double_precision (TRUE, the default, for 64-bit floats; FALSE for 32-bit floats, faster on most GPUs), and arguments of survival::coxph.control() (e.g. iter.max). For the methods: not used.
object	A coxphGPU object.
conf.int	level for computation of the confidence intervals. If set to FALSE no confidence intervals are printed
scale	vector of scale factors for the coefficients, defaults to 1. The printed coefficients, se, and confidence intervals will be associated with one scale unit.

Details

To compute bootstrap confidence intervals with bootstrap(), fit the model with both x = TRUE, which stores the design matrix, and patient_id, which stores the patient of each row used by the model. The bootstrap then resamples exactly the data the model was fitted on.

Value

`coxphGPU()` returns an object of class `coxphGPU`, with the components of a `survival::coxph()` fit (coefficients, variance, log-likelihood, ...). With `patient_id`, it also contains `patient_id` (the column name) and `patient_ids` (the patient of each row used by the model).

`summary()` returns an object of class `summary.coxphGPU`, with confidence intervals for the hazard ratios from the normal approximation and, if `bootstrap()` was called on the model, from the bootstrap. Its main components are:

- `conf.int`: matrix with one row per coefficient and the confidence limits of `exp(coef)`.
- `conf.int_bootstrap`: the same limits, from the bootstrap.
- `logtest`, `sctest`, `waldtest`: likelihood ratio, score and Wald tests of the model.
- `concordance`: concordance statistic and its standard error.
- `rsq`: approximate R^2 of Nagelkerke (Biometrika, 1991).

References

Therneau T (2026). *A Package for Survival Analysis in R*. R package version 3.8-9, https://CRAN.R-project.org/package=survival. `coxphGPU()` follows the interface of its `coxph()`, calls it for the model frame and the martingale residuals, and is tested against it.

Therneau TM, Grambsch PM (2000). *Modeling Survival Data: Extending the Cox Model*. Springer, New York.

See Also

`survival::coxph()`, `bootstrap()`

Examples

```
## Not run:
library(survival)
library(WCE)
data(drugdata)

## Check CUDA drivers (if FALSE you use CPU)
use_cuda()

## Cox Proportional Hazards point estimate; x = TRUE and patient_id are
## required if you plan to call bootstrap() on the result afterwards.
fit <- coxphGPU(Surv(Start, Stop, Event) ~ sex + age,
                data = drugdata, x = TRUE, patient_id = "Id")

## Bootstrap-based inference, as a separate step
fit_boot <- bootstrap(fit, R = 1000,
                      batchsize = if (use_cuda()) 200 else 10)

summary(fit_boot)

## End(Not run)
```

HR *Hazard Ratio of a WCE Model*

Description

Computes, from a model fitted by `wceGPU()`, the hazard ratio between two exposure histories.

Usage

```
HR(object, vecnum, vecdenom, level = 0.95)
```

Arguments

`object`	A `wceGPU` object.
`vecnum`	Numeric vector: the exposure history of interest (numerator of the HR), in reverse chronological order: the first value is the exposure at the current time, the second the exposure one time unit earlier, and so on. A vector longer than the `cutoff` of the model is truncated, and a shorter one is padded with zeros (no exposure further back in time).
`vecdenom`	Numeric vector: the reference exposure history (denominator of the HR), in the same order and with the same length rules as `vecnum`.
`level`	Level of the bootstrap confidence interval of the HR (default `0.95`).

Value

A one-row matrix with the HR and, if `bootstrap()` was called on `object`, the lower and upper limits of its bootstrap percentile confidence interval.

Examples

```
## Not run:
# Dataset
drugdata <- WCE::drugdata

# WCE model with bootstrap (example with 20 bootstraps)
cutoff <- 90
wce_gpu <- wceGPU(data = drugdata, nknots = 1, cutoff = cutoff,
                  id = "Id", event = "Event", start = "Start",
                  stop = "Stop", expos = "dose",
                  covariates = c("age", "sex"),
                  constrained = FALSE, aic = FALSE, confint = 0.95,
                  x = TRUE)
wce_gpu_bootstrap <- bootstrap(wce_gpu, R = 20)

# Exposed at a dose vs. unexposed
exposed   <- rep(1, cutoff)
unexposed <- rep(0, cutoff)
```

```
HR(wce_gpu_bootstrap, exposed, unexposed)

## End(Not run)
```

`predict.coxphGPU` *Predictions from a coxphGPU Model*

Description

Computes fitted values and regression terms for a model fitted by `coxphGPU()`, with `survival::predict.coxph()`.

Usage

```
## S3 method for class 'coxphGPU'
predict(
  object,
  newdata,
  type = c("lp", "risk", "expected", "terms", "survival"),
  se.fit = FALSE,
  na.action = na.pass,
  terms = names(object$assign),
  collapse,
  reference = c("strata", "sample", "zero"),
  ...
)
```

Arguments

`object`	A coxphGPU object.
`newdata`	Optional new data at which to do predictions. If absent, predictions are for the data frame used in the original fit. When coxph has been called with a formula argument created in another context, i.e., coxph has been called within another function and the formula was passed as an argument to that function, there can be problems finding the data set. See the note below.
`type`	the type of predicted value. Choices are the linear predictor (`"lp"`), the risk score exp(lp) (`"risk"`), the expected number of events given the covariates and follow-up time (`"expected"`), and the terms of the linear predictor (`"terms"`). The survival probability for a subject is equal to exp(-expected).
`se.fit`	if TRUE, pointwise standard errors are produced for the predictions.
`na.action`	applies only when the `newdata` argument is present, and defines the missing value action for the new data. The default is to include all observations. When there is no newdata, then the behavior of missing is dictated by the na.action option of the original fit.
`terms`	if type="terms", this argument can be used to specify which terms should be included; the default is all.

collapse	optional vector of subject identifiers. If specified, the output will contain one entry per subject rather than one entry per observation.
reference	reference for centering predictions, see details below
...	For future methods

Value

The predictions computed by survival::predict.coxph(): a numeric vector with one element per observation, or a matrix with one column per model term if type = "terms". If se.fit = TRUE, a list with components fit (the predictions) and se.fit (their standard errors).

References

C Crowson, E Atkinson and T Therneau, Assessing calibration of prognostic risk scores, Stat Methods Med Res, 2016.

See Also

survival::predict.coxph()

Examples

```
## Not run:
library(survival)
old_options <- options(na.action = na.exclude) # retain NA in predictions
fit <- coxphGPU(Surv(time, status) ~ age + ph.ecog + strata(inst), lung)
predict(fit, type = "lp")
predict(fit, type = "expected")
predict(fit, type = "risk", se.fit = TRUE)
predict(fit, type = "terms", se.fit = TRUE)
options(old_options) # restore the previous setting

## End(Not run)
```

---

print.coxphGPU *Print a coxphGPU Model*

Description

Prints a model fitted by coxphGPU() in the same way as a survival::coxph() fit. If bootstrap() was called on the model, a note points to summary() for the bootstrap confidence intervals.

Usage

```
## S3 method for class 'coxphGPU'
print(x, ..., digits = max(1L, getOption("digits") - 3L), signif.stars = FALSE)
```

Arguments

x	A coxphGPU object.
...	Not used.
digits	Number of significant digits to print.
signif.stars	Logical. If TRUE, stars flag the small p-values.

Value

x, invisibly. Called for its side effect: printing the model.

See Also

coxphGPU(), survival::coxph()

---

residuals.coxphGPU *Residuals of a coxphGPU Model*

Description

Calculates martingale, deviance, score or Schoenfeld residuals for a Cox proportional hazards model.

Usage

```
## S3 method for class 'coxphGPU'
residuals(
  object,
  ...,
  type = c("martingale", "deviance", "score", "schoenfeld", "dfbeta", "dfbetas",
    "scaledsch", "partial"),
  collapse = FALSE,
  weighted = (type %in% c("dfbeta", "dfbetas"))
)
```

Arguments

object	an object inheriting from class coxph, representing a fitted Cox regression model. Typically this is the output from the coxph function.
...	other unused arguments
type	character string indicating the type of residual desired. Possible values are "martingale", "deviance", "score", "schoenfeld", "dfbeta"', "dfbetas", "scaledsch" and "partial". Only enough of the string to determine a unique match is required.
collapse	either a logical (TRUE/FALSE), or a vector indicating which rows to collapse (sum) over. In time-dependent models more than one row data can pertain to a single individual or group.

`weighted` if TRUE and the model was fit with case weights, then the weighted residuals are returned.

Value

A vector or matrix of residuals, computed by `survival::residuals.coxph()`:

- martingale and deviance residuals: a numeric vector with one element per observation, or one per group if `collapse` is used;
- score, dfbeta and dfbetas residuals: a matrix with one row per observation and one column per coefficient;
- Schoenfeld and scaled Schoenfeld residuals: a matrix with one row per event and one column per coefficient;
- partial residuals: a matrix with one row per observation and one column per model term.

References

T. Therneau, P. Grambsch, and T. Fleming. "Martingale based residuals for survival models", *Biometrika*, March 1990.

See Also

`survival::residuals.coxph()`

Examples

```
## Not run:
library(survival)
fit <- coxphGPU(Surv(start, stop, event) ~ age + surgery,
                data = heart)

# Martingale residuals
mresid <- resid(fit, collapse = heart$id)

## End(Not run)
```

---

use_cuda *Check Whether a GPU Is Used*

Description

Tells whether computations run on a GPU, i.e. whether 'PyTorch' detects CUDA drivers and a CUDA-capable GPU.

Usage

```
use_cuda()
```

Value

A single logical value: TRUE if 'PyTorch' detects a CUDA-capable GPU, in which case computations run on it by default; FALSE if they run on the CPU. An error is raised if the Python backend cannot be loaded.

Examples

```
## Not run:
use_cuda()

## End(Not run)
```

---

wceGPU *Fast Weighted Cumulative Exposure Model*

Description

Fits a weighted cumulative exposure (WCE) model on the GPU (or the CPU). Only point estimates are computed: call bootstrap() on the result for bootstrap confidence intervals, and inspect it with summary() and plot().

Usage

```
wceGPU(data, nknots, cutoff, constrained = FALSE, aic = FALSE, id,
       event, start, stop, expos, covariates = NULL,
       confint = 0.95, x = FALSE, ...)

## S3 method for class 'wceGPU'
summary(object, allres = FALSE, ...)

## S3 method for class 'wceGPU'
plot(x, ..., hist.covariates = FALSE)

## S3 method for class 'wceGPU'
confint(object, parm, level = 0.95, ..., digits = 3)
```

Arguments

data	Data frame in counting-process (long) format, with one row per subject and time unit.
nknots	Number of interior knots of the cubic spline that models the weight function. With several values (e.g. 1:3), one model is fitted per value and the one with the smallest information criterion (see aic) is kept; best.nknots and the *.grid components of the result describe all the candidates.
cutoff	Length, in time units, of the exposure history that affects the hazard: the weight function is estimated for lags 0 to cutoff - 1 (the cutoff columns of WCEmat), and older exposures have no effect.

`constrained` Shape constraint on the weight function. `FALSE` (default): none. `"Right"` (or `"R"`): it decreases smoothly to zero at the oldest lag. `"Left"` (or `"L"`): it is zero at lag 0, so the current exposure has no effect.

`aic` Logical. Information criterion used to choose among the `nknots` candidates and reported in `info.criterion`: the AIC if `TRUE`, the BIC if `FALSE` (default). Both equal $-2\log(PL) + kp$, with PL the partial likelihood and p the number of estimated parameters; $k = 2$ for the AIC and $k = \log(d)$, with d the number of events, for the BIC (Volinsky and Raftery, 2000).

`id` Name of the column of `data` that identifies the subjects.

`event` Name of the column of `data` with the event indicator, coded `1` (event) or `0` (no event).

`start` Name of the column of `data` with the start time of each interval (the `time` argument of `survival::Surv()`).

`stop` Name of the column of `data` with the end time of each interval (the `time2` argument of `survival::Surv()`).

`expos` Name of the column of `data` with the exposure.

`covariates` Character vector with the names of the columns of `data` to adjust for. `NULL` (default): no adjustment.

`confint` Level of the bootstrap confidence intervals computed by `bootstrap()` (default `0.95`).

`x` For `wceGPU()`: logical. If `TRUE`, the columns of `data` used by the model (`id`, `event`, `start`, `stop`, `expos` and `covariates`) are stored in the result, as `data`. Required by `bootstrap()`, which resamples exactly the data the model was fitted on. For `plot()`: a `wceGPU` object.

`...` For `wceGPU()`: `device` (`"cpu"` or `"cuda"`; default: the GPU if one is available) and `double_precision` (`TRUE`, the default, for 64-bit floats; `FALSE` for 32-bit floats, faster on most GPUs). For the methods: not used.

`object` A `wceGPU` object.

`allres` Logical. If `TRUE`, `summary()` prints every candidate number of knots that was tried, with its partial log-likelihood and information criterion, instead of the summary of the selected model (default `FALSE`).

`hist.covariates` Logical. If `TRUE` and `bootstrap()` was called on the model, `plot()` also draws, for each covariate, the histogram of its bootstrap coefficients (default `FALSE`).

`parm` Covariates whose coefficients get a confidence interval, as numbers or names. Missing (default): all of them.

`level` Confidence level of `confint()` (default `0.95`).

`digits` Number of significant digits of the levels in the column names of `confint()` (default `3`).

Details

At each time t, the exposure enters the Cox model through its weighted cumulative exposure over the last `cutoff` time units,

$$WCE(t) = \sum_{u=0}^{cutoff-1} w(u)X(t-u),$$

where $X(t-u)$ is the exposure u time units before t. The weight function w is a cubic spline with `nknots` interior knots. $WCE(t)$ is therefore a linear combination of spline-weighted exposure histories, which enter a Cox proportional hazards model as time-dependent covariates, together with `covariates`. Their estimated coefficients give w, returned in `WCEmat`.

Value

`wceGPU()` returns an object of class `wceGPU`: a list that describes the selected model, i.e. the candidate in `nknots` that minimizes the information criterion. Its main components are:

- `WCEmat`: matrix with one row and `cutoff` columns, the estimated weight function.
- `beta.hat.covariates`, `se.covariates`: one-row matrices with the estimated coefficients of the covariates and their standard errors.
- `est`, `SED`: one-row matrices with the estimated spline coefficients and their standard errors.
- `vcovmat`: list holding the variance-covariance matrix of all the coefficients (covariates, then spline coefficients).
- `knotsmat`: one-row matrix with the knots of the spline basis.
- `loglik`, `info.criterion`, `nevents`: partial log-likelihood, information criterion (the BIC, or the AIC if `aic = TRUE`) and number of events.
- `best.nknots`: the selected number of interior knots. `nknots.grid`, `info.criterion.grid` and `loglik.grid` give the candidates and the information criterion and partial log-likelihood obtained for each.
- `data`: with `x = TRUE`, the columns of `data` used by the model.

After `bootstrap()`, the object also contains:

- `WCEmat_bootstrap`, `bootstrap_beta.hat.covariates`: matrices with one row per bootstrap replicate, for the weight function and for the coefficients of the covariates.
- `WCEmat_CI`, `coef_CI`: two-row matrices with their lower and upper percentile confidence limits, at level `confint`.
- `bootstrap.best.nknots`: the number of knots selected in each replicate.
- `bootstrap_est`, `est_CI`: the same for the spline coefficients, only when a single `nknots` value is given.

`summary()` and `plot()` are called for their side effects (printing the model, plotting the weight function) and return `NULL` invisibly. `confint()` returns a matrix with one row per covariate and two columns: the lower and upper confidence limits of its coefficient, based on the normal approximation.

References

Sylvestre MP, Abrahamowicz M. Flexible modeling of the cumulative effects of time-dependent exposures on the hazard. Stat Med. 2009 Nov 30;28(27):3437-53. doi:10.1002/sim.3701

Sylvestre M (2025). *WCE: Weighted Cumulative Exposure Models*. R package version 1.0.4. doi:10.32614/CRAN.package.WCE. The reference implementation of the method: `wceGPU()` follows its interface, and its results are tested against it.

Volinsky CT, Raftery AE (2000). Bayesian information criterion for censored survival models. Biometrics, 56(1), 256-262. doi:10.1111/j.0006341X.2000.00256.x

See Also

`bootstrap()`

Examples

```
## Not run:
# Dataset
drugdata <- WCE::drugdata

# WCE model, point estimate only
wce_gpu <- wceGPU(data = drugdata, nknots = 1, cutoff = 90, id = "Id",
                  event = "Event", start = "Start", stop = "Stop",
                  expos = "dose", covariates = c("age", "sex"),
                  constrained = FALSE, aic = FALSE, confint = 0.95,
                  x = TRUE)

# Results
wce_gpu
summary(wce_gpu)

# See estimated weight function
plot(wce_gpu)

# Bootstrap-based inference, as a separate step (example with 20
# replicates, but normally R > 500)
wce_gpu_bootstrap <- bootstrap(wce_gpu, R = 20)

# See confidence bands for the estimated weight function due to bootstrap
plot(wce_gpu_bootstrap)

# All estimated coefficients in bootstrap
coef(wce_gpu_bootstrap)

# Estimate a HR (Exposed at a dose vs. unexposed)
exposed   <- rep(1, 90)
unexposed <- rep(0, 90)

HR(wce_gpu_bootstrap, exposed, unexposed)

## End(Not run)
```

Index

bootstrap, 3
bootstrap(), *2*, *5–8*, *10*, *13–16*

confint.wceGPU *(*wceGPU*)*, 13
coxph.control, *6*
coxphGPU, 5
coxphGPU(), *2, 3*, *9–11*

HR, 8
HR(), *2*, *5*

plot.wceGPU *(*wceGPU*)*, 13
predict.coxphGPU, 9
print.coxphGPU, 10

residuals.coxphGPU, 11

summary.coxphGPU *(*coxphGPU*)*, 5
summary.wceGPU *(*wceGPU*)*, 13
survival::coxph(), *5*, *7*, *10, 11*
survival::coxph.control(), *4*, *6*
survival::predict.coxph(), *9, 10*
survival::residuals.coxph(), *12*
survival::Surv(), *6*, *14*
survivalGPU *(*survivalGPU-package*)*, 2
survivalGPU-package, 2

use_cuda, 12
use_cuda(), *2*

wceGPU, 13
wceGPU(), *2, 3*, *8*