Package: bigPLScox (via r-universe)

October 23, 2026

Type Package

Version 0.8.1

Date 2025-11-15

Depends R (>= 4.0.0)

Imports bigmemory, bigalgebra, bigSurvSGD, caret, doParallel, foreach, kernlab, methods, Rcpp, risksetROC, rms, sgPLS, survAUC, survcomp, survival

LinkingTo BH, Rcpp, RcppArmadillo, bigmemory

Suggests bench, knitr, plsRcox, mvtnorm, readr, rmarkdown, testthat (>= 3.0.0)

Title Partial Least Squares for Cox Models with Big Matrices

Author Frederic Bertrand [cre, aut] (ORCID: <https://orcid.org/0000-0002-0837-8281>), Myriam Maumy-Bertrand [aut] (ORCID: <https://orcid.org/0000-0002-4615-1512>)

Maintainer Frederic Bertrand <frederic.bertrand@lecnam.net>

Description Provides Partial least squares Regression and various regular, sparse or kernel, techniques for fitting Cox models for big data. Provides a Partial Least Squares (PLS) algorithm adapted to Cox proportional hazards models that works with 'bigmemory' matrices without loading the entire dataset in memory. Also implements a gradient-descent based solver for Cox proportional hazards models that works directly on 'bigmemory' matrices. Bertrand and Maumy (2023) <https://hal.science/hal-05352069>, and <https://hal.science/hal-05352061> highlighted fitting and cross-validating PLS-based Cox models to censored big data.

License GPL-3

Encoding UTF-8

URL https://fbertran.github.io/bigPLScox/, https://github.com/fbertran/bigPLScox

BugReports https://github.com/fbertran/bigPLScox/issues

Classification/MSC 62N01, 62N02, 62N03, 62N99

RoxygenNote 7.3.3

VignetteBuilder knitr

Config/testthat/edition 3

SystemRequirements C++17

NeedsCompilation yes

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

Date/Publication 2025-11-18 14:40:06 UTC

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

RemoteRef HEAD

RemoteSha da18fcc84132ab3ac2594cbc485b4f4108d49cbd

Contents

bigPLScox-package . . . 3
align_big_plscox . . . 4
big_pls_cox . . . 5
big_pls_cox_fast . . . 6
big_pls_cox_gd . . . 7
big_pls_cox_transform . . . 10
bigmatrix-operations . . . 11
bigscale . . . 12
bigSurvSGD.na.omit . . . 14
component_information . . . 17
computeDR . . . 18
cox_deviance_residuals . . . 21
coxDKgplsDR . . . 22
coxDKsgplsDR . . . 27
coxDKspls_sgplsDR . . . 32
coxgpls . . . 37
coxgplsDR . . . 41
coxsgpls . . . 45
coxsgplsDR . . . 49
coxspls_sgpls . . . 53
coxspls_sgplsDR . . . 57
cv.big_pls_cox . . . 61
cv.coxDKgplsDR . . . 63
cv.coxDKsgplsDR . . . 67
cv.coxDKspls_sgplsDR . . . 72
cv.coxgpls . . . 77
cv.coxgplsDR . . . 81
cv.coxsgpls . . . 86
cv.coxsgplsDR . . . 91
cv.coxspls_sgpls . . . 95

cv.coxspls_sgplsDR 100
dataCox 105
dCox_sim 106
gd_diagnostics 106
micro.censure 107
partialbigSurvSGDv0 109
plot.big_pls_cox 111
plot.big_pls_cox_gd 111
predict.big_pls_cox 112
predict.big_pls_cox_fast 113
predict.big_pls_cox_gd 114
predict_cox_pls 114
predict_pls_latent 117
print.big_pls_cox 119
print.big_pls_cox_gd 120
print.summary.big_pls_cox_fast 120
sim_data 121
summary.big_pls_cox 122
summary.big_pls_cox_fast 122
summary.big_pls_cox_gd 123
Xmicro.censure_compl_imp 123

Index **126**

`bigPLScox-package` *bigPLScox-package*

Description

Provides Partial least squares Regression for regular, generalized linear and Cox models for big data. It allows for missing data in the explanatory variables. Repeated k-fold cross-validation of such models using various criteria. Bootstrap confidence intervals constructions are also available.

Author(s)

Maintainer: Frederic Bertrand <frederic.bertrand@lecnam.net> (ORCID)

Authors:

- Myriam Maumy-Bertrand <myriam.maumy@ehesp.fr> (ORCID)

References

Maumy, M., Bertrand, F. (2023). PLS models and their extension for big data. Joint Statistical Meetings (JSM 2023), Toronto, ON, Canada.

Maumy, M., Bertrand, F. (2023). bigPLS: Fitting and cross-validating PLS-based Cox models to censored big data. BioC2023 — The Bioconductor Annual Conference, Dana-Farber Cancer Institute, Boston, MA, USA. Poster. https://doi.org/10.7490/f1000research.1119546.1

Bastien, P., Bertrand, F., Meyer, N., and Maumy-Bertrand, M. (2015). Deviance residuals-based sparse PLS and sparse kernel PLS for binary classification and survival analysis. *BMC Bioinformatics*, 16, 211.

See Also

`big_pls_cox()` and `big_pls_cox_gd()`

Examples

```
set.seed(314)
library(bigPLScox)
data(sim_data)
head(sim_data)
```

`align_big_plscox` *Align a GD fit to a PLS fit (optional refit)*

Description

Align a GD fit to a PLS fit (optional refit)

Usage

```
align_big_plscox(fit_gd, fit_pls, time, status, rotate = TRUE, refit = TRUE)
```

Arguments

`fit_gd`	object from big_pls_cox_gd()
`fit_pls`	object from big_pls_cox()
`time, status`	Surv parts used for refit
`rotate`	logical; Procrustes-rotate GD scores to the PLS basis
`refit`	logical; refit a Cox model on the rotated GD scores

Value

fit_gd with $scores/$coefficients/$cox_fit possibly updated

`big_pls_cox` *Partial Least Squares Components for Cox Models with Big Matrices*

Description

Compute Partial Least Squares (PLS) components tailored for Cox proportional hazards models when predictors are stored as a `big.matrix` from the **bigmemory** package.

Usage

```
big_pls_cox(
  X,
  time,
  status,
  ncomp = 2L,
  control = survival::coxph.control(),
  keepX = NULL
)
```

Arguments

`X`	A numeric matrix or a `bigmemory::big.matrix` object containing the predictors.
`time`	Numeric vector of survival times.
`status`	Integer (0/1) vector of event indicators.
`ncomp`	Number of latent components to compute.
`control`	Optional list passed to `survival::coxph.control`.
`keepX`	Optional integer vector specifying the number of variables to retain (naive sparsity) in each component. A value of zero keeps all predictors. If a single integer is supplied it is recycled across components.

Details

The function standardises each predictor column, iteratively builds latent scores using martingale residuals from Cox fits, and deflates the predictors without materialising the full design matrix in memory. Both in-memory and file-backed **bigmemory** matrices are supported.

Value

A list with the computed scores, loadings, weights, scaling information and the fitted Cox model returned by `survival::coxph.fit`.

References

Maumy, M., Bertrand, F. (2023). PLS models and their extension for big data. Joint Statistical Meetings (JSM 2023), Toronto, ON, Canada.

Maumy, M., Bertrand, F. (2023). bigPLS: Fitting and cross-validating PLS-based Cox models to censored big data. BioC2023 — The Bioconductor Annual Conference, Dana-Farber Cancer Institute, Boston, MA, USA. Poster. https://doi.org/10.7490/f1000research.1119546.1

Bastien, P., Bertrand, F., Meyer, N., & Maumy-Bertrand, M. (2015). Deviance residuals-based sparse PLS and sparse kernel PLS for censored data. *Bioinformatics*, 31(3), 397–404. doi:10.1093/bioinformatics/btu660

Bertrand, F., Bastien, P., Meyer, N., & Maumy-Bertrand, M. (2014). PLS models for censored data. In *Proceedings of UseR! 2014* (p. 152).

See Also

`big_pls_cox_gd()`, `predict.big_pls_cox()`, `select_ncomp()`, `computeDR()`.

Examples

```
if (requireNamespace("survival", quietly = TRUE)) {
  set.seed(1)
  X <- matrix(rnorm(100), nrow = 20)
  time <- rexp(20)
  status <- rbinom(20, 1, 0.5)
  fit <- big_pls_cox(X, time, status, ncomp = 2)
  str(fit)
}
```

---

big_pls_cox_fast *Partial Least Squares Components for Cox Models (fast backend)*

Description

Compute PLS components for Cox models, using a fast C++ backend for both in-memory matrices and `bigmemory::big.matrix` objects.

Usage

```
big_pls_cox_fast(
  X,
  time,
  status,
  ncomp = 2L,
  control = survival::coxph.control(),
  keepX = NULL
)
```

Arguments

X	A numeric matrix or a bigmemory::big.matrix object containing the predictors.
time	Numeric vector of survival times.
status	Integer (0/1) vector of event indicators.
ncomp	Number of latent components to compute.
control	Optional list passed to survival::coxph.control.
keepX	Optional integer vector specifying the number of variables to retain (naive sparsity) in each component. A value of zero keeps all predictors. If a single integer is supplied it is recycled across components.

Value

A list with the computed scores, loadings, weights, scaling information and the fitted Cox model returned by survival::coxph.fit.

References

Maumy, M., Bertrand, F. (2023). PLS models and their extension for big data. Joint Statistical Meetings (JSM 2023), Toronto, ON, Canada.

Maumy, M., Bertrand, F. (2023). bigPLS: Fitting and cross-validating PLS-based Cox models to censored big data. BioC2023 — The Bioconductor Annual Conference, Dana-Farber Cancer Institute, Boston, MA, USA. Poster. https://doi.org/10.7490/f1000research.1119546.1

Bastien, P., Bertrand, F., Meyer, N., & Maumy-Bertrand, M. (2015). Deviance residuals-based sparse PLS and sparse kernel PLS for censored data. *Bioinformatics*, 31(3), 397–404. doi:10.1093/bioinformatics/btu660

Bertrand, F., Bastien, P., Meyer, N., & Maumy-Bertrand, M. (2014). PLS models for censored data. In *Proceedings of UseR! 2014* (p. 152).

---

big_pls_cox_gd *Gradient based PLS Cox for big matrices*

Description

Fit a PLS Cox model where the PLS components are computed once and the Cox regression in the latent space is optimised by gradient based methods.

This function is intended as a faster, approximate alternative to big_pls_cox_fast() when many fits are required or when the design is stored as a bigmemory::big.matrix.

Usage

```
big_pls_cox_gd(
  X,
  time,
  status,
  ncomp = 2L,
  max_iter = 2000L,
  tol = 1e-08,
  learning_rate = 0.05,
  keepX = NULL,
  coxfit = TRUE,
  method = c("gd", "bb", "nesterov", "bfgs"),
  diag = TRUE
)
```

Arguments

X	A bigmemory::big.matrix containing the design matrix (rows are observations).
time	A numeric vector of follow-up times with length equal to the number of rows of X.
status	A numeric or integer vector of the same length as time containing the event indicators (1 for an event, 0 for censoring).
ncomp	An integer giving the number of components (columns) to use from X. Defaults to min(5, ncol(X)).
max_iter	Maximum number of gradient iterations.
tol	Convergence tolerance for the optimisation in the Cox space. Both the change in log-likelihood and the Euclidean change in the coefficient vector are monitored.
learning_rate	Initial learning rate for first order methods. This is used by "gd" and as a starting scale for "bb" and "nesterov". It is ignored by "bfgs".
keepX	Optional integer vector describing the number of predictors to retain per component (naive sparsity). A value of zero keeps all predictors.
coxfit	Optional Boolean to fit a Cox model on the extracted components.
method	Optimisation scheme used in the latent space. One of "gd" Simple fixed step gradient descent. This is the most transparent method and is useful for debugging and didactic purposes. "bb" Barzilai Borwein step size. Uses a diagonal quasi-second-order update of the step size based on the last two gradients. Often converges faster than "gd" at similar computational cost. "nesterov" Nesterov type accelerated gradient with a fixed momentum schedule. Can yield smoother convergence in early iterations, at the price of slightly more bookkeeping. "bfgs" Quasi Newton update in the latent space, with a limited memory BFGS type approximation of the Hessian. This gives the most accurate coefficients in a small number of iterations but requires more linear algebra per step.

The default is "bb", which provides a good balance between speed and robustness in most examples.

`diag` Logical. If TRUE, store iteration level diagnostics.

Details

The function first computes PLS components using the same procedure as `big_pls_cox_fast()`, then holds these components fixed and optimises the Cox partial log-likelihood in the reduced space.

The coefficients stored in `fit$coefficients` are the result of the chosen optimisation method and are approximate. The field `fit$cox_fit` contains the Cox model refitted by `survival::coxph()` on the final PLS scores. Prediction functions use the coefficients from `cox_fit` so that linear predictors are directly interpretable as Cox risk scores.

The optimisation tolerances control the compromise between speed and accuracy. If you need very close agreement with the exact PLS Cox solution, use `big_pls_cox_fast()`. If you only need accurate risk rankings and want to fit many models, the gradient based method with "bb" or "bfgs" is usually sufficient.

Value

An object of class "big_pls_cox_gd" that contains:

- `coefficients`: Estimated Cox regression coefficients on the latent scores.
- `loglik`: Final partial log-likelihood value.
- `iterations`: Number of gradient-descent iterations performed.
- `converged`: Logical flag indicating whether convergence was achieved.
- `scores`: Matrix of latent score vectors (one column per component).
- `loadings`: Matrix of loading vectors associated with each component.
- `weights`: Matrix of PLS weight vectors.
- `center`: Column means used to centre the predictors.
- `scale`: Column scales (standard deviations) used to standardise the predictors.
- `keepX`: Vector describing the number of predictors retained per component.
- `time`: Numeric vector of follow-up times.
- `status`: Numeric or integer vector containing the event indicators.
- `loglik_trace`: Trace of the loglik.
- `step_trace`: Trace of the steps
- `gradnorm_trace`: Trace of the gradnorm.
- `cox_fit`: Final Cox model fitted on the components.

References

Maumy, M., Bertrand, F. (2023). PLS models and their extension for big data. Joint Statistical Meetings (JSM 2023), Toronto, ON, Canada.

Maumy, M., Bertrand, F. (2023). bigPLS: Fitting and cross-validating PLS-based Cox models to censored big data. BioC2023 — The Bioconductor Annual Conference, Dana-Farber Cancer Institute, Boston, MA, USA. Poster. https://doi.org/10.7490/f1000research.1119546.1

Bastien, P., Bertrand, F., Meyer, N., & Maumy-Bertrand, M. (2015). Deviance residuals-based sparse PLS and sparse kernel PLS for censored data. *Bioinformatics*, 31(3), 397–404. doi:10.1093/bioinformatics/btu660

Bertrand, F., Bastien, P., Meyer, N., & Maumy-Bertrand, M. (2014). PLS models for censored data. In *Proceedings of UseR! 2014* (p. 152).

See Also

`big_pls_cox()`, `predict.big_pls_cox()`, `select_ncomp()`, `computeDR()`.

Examples

```
library(bigmemory)
set.seed(1)
n <- 50
p <- 10
X <- bigmemory::as.big.matrix(matrix(rnorm(n * p), n, p))
time <- rexp(n, rate = 0.1)
status <- rbinom(n, 1, 0.7)
fit <- big_pls_cox_gd(X, time, status, ncomp = 3, max_iter = 200)
str(fit)
head(fit$scores)
```

---

big_pls_cox_transform *Transform new data to PLS–Cox scores*

Description

Project new observations onto previously fitted PLS–Cox components.

Usage

```
big_pls_cox_transform(
  X,
  means,
  sds,
  weights,
  loadings,
  comps = seq_len(ncol(weights))
)
```

Arguments

X	New data: a numeric matrix or a bigmemory::big.matrix.
means	Column means used to center the original predictors.
sds	Column standard deviations used to scale the original predictors.
weights	PLS weight matrix (p x ncomp) from a fitted model.
loadings	PLS loading matrix (p x ncomp) from a fitted model.
comps	Integer vector of component indices to return (1-based).

Value

A numeric matrix of scores with one row per observation in X and one column per requested component.

---

bigmatrix-operations *Matrix and arithmetic operations for big.matrix objects*

Description

These methods extend the base matrix multiplication operator (%*%) and the group generic Arithmetic so that big.matrix objects can interoperate with base R matrices and numeric scalars using the high-performance routines provided by **bigalgebra**.

Usage

```
## S4 method for signature 'big.matrix,big.matrix'
x %*% y

## S4 method for signature 'matrix,big.matrix'
x %*% y

## S4 method for signature 'big.matrix,matrix'
x %*% y

## S4 method for signature 'big.matrix,big.matrix'
Arith(e1, e2)

## S4 method for signature 'big.matrix,matrix'
Arith(e1, e2)

## S4 method for signature 'matrix,big.matrix'
Arith(e1, e2)

## S4 method for signature 'numeric,big.matrix'
Arith(e1, e2)

## S4 method for signature 'big.matrix,numeric'
Arith(e1, e2)
```

Arguments

x, y	Matrix operands supplied either as big.matrix instances or base R matrices, depending on the method signature.
e1, e2	Numeric operands, which may be big.matrix objects, base R matrices, or numeric scalars depending on the method signature.

Details

Matrix multiplications dispatch to bigalgebra::dgemm(), mixed arithmetic on matrices relies on bigalgebra::daxpy(), and scalar/matrix combinations use bigalgebra::dadd() when appropriate.

See Also

bigmemory::big.matrix(), bigalgebra::dgemm(), bigalgebra::daxpy(), bigalgebra::dadd()

Examples

```
if (requireNamespace("bigmemory", quietly = TRUE) &&
    requireNamespace("bigalgebra", quietly = TRUE)) {
  x <- bigmemory::big.matrix(2, 2, init = 1)
  y <- bigmemory::big.matrix(2, 2, init = 2)
  x %*% y
  x + y
  x * 3
}
```

---

bigscale *Construct Scaled Design Matrices for Big Survival Models*

Description

Prepares a large-scale feature matrix for stochastic gradient descent byapplying optional normalisation, stratified sampling, and batching rules.

Usage

```
bigscale(
  formula = survival::Surv(time = time, status = status) ~ .,
  data,
  norm.method = "standardize",
  strata.size = 20,
  batch.size = 1,
  features.mean = NULL,
  features.sd = NULL,
  parallel.flag = FALSE,
```

```
  num.cores = NULL,
  bigmemory.flag = FALSE,
  num.rows.chunk = 1e+06,
  col.names = NULL,
  type = "short"
)
```

Arguments

formula	formula used to extract the outcome and predictors that should be included in the scaled design matrix.
data	Input data source containing the variables referenced in formula.
norm.method	Normalisation strategy (for example centring or standardising columns) applied to the feature matrix.
strata.size	Number of observations to retain from each stratum when constructing stratified batches.
batch.size	Total size of each mini-batch produced by the scaling routine.
features.mean	Optional vector of column means that can be reused to normalise multiple data sets in a consistent manner.
features.sd	Optional vector of column standard deviations that pairs with features.mean during scaling.
parallel.flag	Logical flag signalling whether the scaling work should be parallelised across cores.
num.cores	Number of processor cores allocated when parallel.flag is TRUE.
bigmemory.flag	Logical flag specifying whether intermediate results should be stored in **bigmemory**-backed matrices.
num.rows.chunk	Chunk size used when streaming data from on-disk objects into memory.
col.names	Optional character vector assigning column names to the generated design matrix.
type	Type of model or preprocessing target being prepared, such as survival or regression.

Value

A scaled design matrix of the scaler class along with metadata describing the transformation that was applied. time.indices: indices of the time variable cens.indices: indices of the censored variables features.indices: indices of the features time.sd: standard deviation of the time variable time.mean: mean of the time variable features.sd: standard deviation of the features features.mean: mean of the features nr: number of rows nc: number of columns col.names: columns names

See Also

bigSurvSGD.na.omit() for fitting models that use the scaled features.

Examples

```
data(micro.censure, package = "bigPLScox")
surv_data <- stats::na.omit(
  micro.censure[, c("survyear", "DC", "sexe", "Agediag")]
)
scaled <- bigscale(
  survival::Surv(survyear, DC) ~ .,
  data = surv_data,
  norm.method = "standardize",
  batch.size = 16
)
```

`bigSurvSGD.na.omit` *Fit Survival Models with Stochastic Gradient Descent*

Description

Performs stochastic gradient descent optimisation for large-scale survival models after removing observations with missing values.

Usage

```
bigSurvSGD.na.omit(
  formula = survival::Surv(time = time, status = status) ~ .,
  data,
  norm.method = "standardize",
  features.mean = NULL,
  features.sd = NULL,
  opt.method = "AMSGrad",
  beta.init = NULL,
  beta.type = "averaged",
  lr.const = 0.12,
  lr.tau = 0.5,
  strata.size = 20,
  batch.size = 1,
  num.epoch = 100,
  b1 = 0.9,
  b2 = 0.99,
  eps = 1e-08,
  inference.method = "plugin",
  num.boot = 1000,
  num.epoch.boot = 100,
  boot.method = "SGD",
  lr.const.boot = 0.12,
  lr.tau.boot = 0.5,
  num.sample.strata = 1000,
```

```
  sig.level = 0.05,
  beta0 = 0,
  alpha = NULL,
  lambda = NULL,
  nlambda = 100,
  num.strata.lambda = 10,
  lambda.scale = 1,
  parallel.flag = FALSE,
  num.cores = NULL,
  bigmemory.flag = FALSE,
  num.rows.chunk = 1e+06,
  col.names = NULL,
  type = "float"
)
```

Arguments

formula	Model formula describing the survival outcome and the set of predictors to include in the optimisation.
data	Input data set or connection to a big-memory backed design matrix that contains the variables referenced in formula.
norm.method	Normalization strategy applied to the feature matrix before optimisation, for example centring or standardising columns.
features.mean	Optional pre-computed column means used when normalising the features so that repeated fits can reuse shared statistics.
features.sd	Optional pre-computed column standard deviations used in concert with features.mean for scaling the predictors.
opt.method	Gradient based optimisation routine to employ, such as vanilla SGD or adaptive methods like Adam.
beta.init	Vector of starting values for the regression coefficients supplied when warm-starting the optimisation.
beta.type	Indicator controlling how beta.init is interpreted, for example whether the coefficients correspond to the original or normalised scale.
lr.const	Base learning-rate constant used by the stochastic gradient descent routine.
lr.tau	Learning-rate decay horizon or damping factor that moderates the step size schedule.
strata.size	Number of observations drawn per stratum when building mini-batches for the optimisation loop.
batch.size	Total number of observations assembled into each stochastic gradient batch.
num.epoch	Number of passes over the training data used during the optimisation.
b1	First exponential moving-average rate used by adaptive methods such as Adam to smooth gradients.
b2	Second exponential moving-average rate used by adaptive methods to smooth squared gradients.

`eps`	Numerical stabilisation constant added to denominators when updating the adaptive moments.
`inference.method`	Inference approach requested after fitting, for example naive asymptotics or bootstrap resampling.
`num.boot`	Number of bootstrap replicates to draw when `inference.method` relies on resampling.
`num.epoch.boot`	Number of optimisation epochs to run within each bootstrap replicate.
`boot.method`	Type of bootstrap scheme to apply, such as ordinary or stratified resampling.
`lr.const.boot`	Learning-rate constant used during bootstrap refits.
`lr.tau.boot`	Learning-rate decay factor applied during bootstrap refits.
`num.sample.strata`	Number of strata sampled without replacement during each bootstrap iteration when stratified resampling is selected.
`sig.level`	Significance level used when constructing confidence intervals or hypothesis tests.
`beta0`	Optional vector of coefficients under the null hypothesis when performing hypothesis tests.
`alpha`	Elastic-net mixing parameter controlling the relative weight of ℓ_1 and ℓ_2 regularisation penalties.
`lambda`	Sequence of regularisation strengths supplied explicitly for penalised estimation.
`nlambda`	Number of automatically generated `lambda` values when a grid is produced internally.
`num.strata.lambda`	Number of strata used when tuning `lambda` via cross-validation or other search procedures.
`lambda.scale`	Scale on which the `lambda` grid is generated, for example logarithmic or linear spacing.
`parallel.flag`	Logical flag enabling parallel computation of gradients or bootstrap replicates.
`num.cores`	Number of processing cores to use when parallel execution is enabled.
`bigmemory.flag`	Logical flag indicating whether intermediate matrices should be stored using **bigmemory** backed objects.
`num.rows.chunk`	Row chunk size to use when streaming data from an on-disk matrix representation.
`col.names`	Optional character vector of column names associated with the feature matrix.
`type`	Type of survival model to fit, for example Cox proportional hazards or accelerated failure time variants.

Value

A fitted model object storing the learned coefficients, optimisation metadata, and any requested inference summaries. coef: Log of hazards ratio. If no inference is used, it returns a vector for estimated coefficients: If inference is used, it returns a matrix including estimates and confidence

intervals of coefficients. In case of penalization, it resturns a matrix with columns corresponding to lambdas. coef.exp: Exponentiated version of coef (hazards ratio). lambda: Returns lambda(s) used for penalizarion. alpha: Returns alpha used for penalizarion. features.mean: Returns means of features, if given or calculated features.sd: Returns standard deviations of features, if given or calculated.

See Also

See Also bigSurvSGD, bigscale for constructing normalised design matrices and partialbigSurvSGDv0 for partial fitting pipelines.

Examples

```
data(micro.censure, package = "bigPLScox")
surv_data <- stats::na.omit(micro.censure[, c("survyear", "DC", "sexe", "Agediag")])
# Increase num.epoch and num.boot for real use
fit <- bigSurvSGD.na.omit(
   survival::Surv(survyear, DC) ~ .,
   data = surv_data,
   norm.method = "standardize",
   opt.method = "adam",
   batch.size = 16,
   num.epoch = 2,
 )
```

---

component_information *Information criteria for component selection*

Description

Computes log-likelihood, AIC and BIC values for nested models using the latent components estimated by big_pls_cox() or big_pls_cox_gd().

Usage

```
component_information(object, max_comp = ncol(object$scores))

## S3 method for class 'big_pls_cox'
component_information(object, max_comp = ncol(object$scores))

## S3 method for class 'big_pls_cox_gd'
component_information(object, max_comp = ncol(object$scores))

select_ncomp(object, criterion = c("AIC", "BIC", "loglik"), ...)
```

Arguments

object	A fitted object of class big_pls_cox or big_pls_cox_gd.
max_comp	Maximum number of components to consider. Defaults to all components stored in the model.
criterion	Criterion to optimise: "AIC", "BIC" or "loglik".
...	Passed to component_information().

Value

A data frame with columns ncomp, loglik, AIC, and BIC.

A list with the table of information criteria and the recommended number of components.

---

computeDR *Compute deviance residuals*

Description

This function computes deviance residuals from a null Cox model. By default it delegates to survival::coxph(), but a high-performance C++ engine is also available for large in-memory or bigmemory::big.matrix design matrices.

Usage

```
computeDR(
  time,
  time2,
  event,
  type,
  origin,
  typeres = "deviance",
  collapse,
  weighted,
  scaleY = TRUE,
  plot = FALSE,
  engine = c("survival", "cpp", "qcpp"),
  method = c("efron", "breslow"),
  X = NULL,
  coef = NULL,
  eta = NULL,
  center = NULL,
  scale = NULL
)
```

Arguments

time	for right censored data, this is the follow up time. For interval data, the first argument is the starting time for the interval.
time2	The status indicator, normally 0=alive, 1=dead. Other choices are TRUE/FALSE (TRUE = death) or 1/2 (2=death). For interval censored data, the status indicator is 0=right censored, 1=event at time, 2=left censored, 3=interval censored. Although unusual, the event indicator can be omitted, in which case all subjects are assumed to have an event.
event	ending time of the interval for interval censored or counting process data only. Intervals are assumed to be open on the left and closed on the right, (start, end]. For counting process data, event indicates whether an event occurred at the end of the interval.
type	character string specifying the type of censoring. Possible values are "right", "left", "counting", "interval", or "interval2". The default is "right" or "counting" depending on whether the time2 argument is absent or present, respectively.
origin	for counting process data, the hazard function origin. This option was intended to be used in conjunction with a model containing time dependent strata in order to align the subjects properly when they cross over from one strata to another, but it has rarely proven useful.
typeres	character string indicating the type of residual desired. Possible values are "martingale", "deviance", "score", "schoenfeld", "dfbeta", "dfbetas", and "scaledsch". Only enough of the string to determine a unique match is required.
collapse	vector indicating which rows to collapse (sum) over. In time-dependent models more than one row data can pertain to a single individual. If there were 4 individuals represented by 3, 1, 2 and 4 rows of data respectively, then collapse=c(1,1,1,2,3,3,4,4,4,4) could be used to obtain per subject rather than per observation residuals.
weighted	if TRUE and the model was fit with case weights, then the weighted residuals are returned.
scaleY	Should the time values be standardized ?
plot	Should the survival function be plotted ?
engine	Either "survival" (default) to call survival::coxph() or "cpp" to use the C++ implementation.
method	Tie handling to use with engine = "cpp": either "efron" (default) or "breslow".
X	Optional design matrix used to compute the linear predictor when engine = "cpp". Supports base matrices, data frames, and bigmemory::big.matrix objects.
coef	Optional coefficient vector associated with X when engine = "cpp".
eta	Optional precomputed linear predictor passed directly to the C++ engine.
center, scale	Optional centring and scaling vectors applied to X before computing the linear predictor with the C++ engine.

Value

Residuals from a null model fit. When `engine = "cpp"`, the returned vector has attributes `"martingale"`, `"cumhaz"`, and `"linear_predictor"`.

Author(s)

Frédéric Bertrand
<frederic.bertrand@lecnam.net>
https://fbertran.github.io/homepage/

References

Bastien, P., Bertrand, F., Meyer, N., and Maumy-Bertrand, M. (2015). Deviance residuals-based sparse PLS and sparse kernel PLS for binary classification and survival analysis. *BMC Bioinformatics*, 16, 211.

Therneau, T.M., Grambsch, P.M. (2000). *Modeling Survival Data: Extending the Cox Model*. Springer.

See Also

coxph

Examples

```
data(micro.censure, package = "bigPLScox")

Y_train_micro <- micro.censure$survyear[1:80]
C_train_micro <- micro.censure$DC[1:80]

Y_DR <- computeDR(Y_train_micro,C_train_micro)
Y_DR <- computeDR(Y_train_micro,C_train_micro,plot=TRUE)

Y_cpp <- computeDR(
  Y_train_micro,
  C_train_micro,
  engine = "cpp",
  eta = rep(0, length(Y_train_micro))
)

Y_qcpp <- computeDR(
  Y_train_micro,
  C_train_micro,
  engine = "qcpp"
)
```

cox_deviance_residuals *Cox deviance residuals via C++ backends*

Description

Compute martingale and deviance residuals for Cox models without materialising intermediate survival fits in R. The functions rely on dedicated C++ implementations that operate either on in-memory vectors or on `bigmemory::big.matrix` objects to enable streaming computations on large datasets.

Usage

```
cox_deviance_residuals(time, status, weights = NULL)

cox_deviance_details(time, status, weights = NULL)

cox_deviance_residuals_big(X, time_col, status_col, weights = NULL)

cox_partial_deviance_big(X, coef, time, status)

benchmark_deviance_residuals(time, status, iterations = 25, methods = list())
```

Arguments

`time`	Numeric vector of follow-up times.
`status`	Numeric or integer vector of the same length as `time` giving the event indicators (1 for an event, 0 for censoring).
`weights`	Optional non-negative case weights. When supplied they must have the same length as `time`.
`X`	A `bigmemory::big.matrix` storing the survival information column-wise.
`time_col, status_col`	Integer indices pointing to the columns of `X` that contain the follow-up time and event indicator respectively.
`coef`	Numeric vector of regression coefficients used to evaluate the partial log-likelihood and deviance on a `big.matrix` design.
`iterations`	Number of iterations used by `bench::mark` when benchmarking the residual computations.
`methods`	Optional named list of alternative residual implementations to compare against in `benchmark_deviance_residuals`.

Details

- `cox_deviance_residuals()` operates on standard R vectors and matches the output of `residuals(coxph(...), type = "deviance")` for right-censored data without ties.

- `cox_deviance_residuals_big()` keeps the computation in C++ while reading directly from a `big.matrix`, avoiding extra copies.
- `cox_partial_deviance_big()` evaluates the partial log-likelihood and deviance for a given coefficient vector and big design matrix. This is useful when selecting the number of latent components via information criteria.

`benchmark_deviance_residuals()` compares the dedicated C++ implementation against reference approaches (for example, the `survival` package) using `bench::mark`. The function returns a tibble with iteration statistics.

Value

- `cox_deviance_residuals()` and `cox_deviance_residuals_big()` return a numeric vector of deviance residuals.
- `cox_deviance_details()` returns a list with cumulative hazard, martingale, and deviance residuals.
- `cox_partial_deviance_big()` returns a list containing the partial log-likelihood, deviance, and the evaluated linear predictor.
- `benchmark_deviance_residuals()` returns a `tibble::tibble`.

Examples

```
if (requireNamespace("survival", quietly = TRUE)) {
  set.seed(123)
  time <- rexp(50)
  status <- rbinom(50, 1, 0.6)
  dr_cpp <- cox_deviance_residuals(time, status)
  dr_surv <- residuals(survival::coxph(survival::Surv(time, status) ~ 1),
                       type = "deviance")
  all.equal(unname(dr_cpp), unname(dr_surv), tolerance = 1e-6)
}
```

---

coxDKgplsDR *Fitting a Direct Kernel group PLS model on the (Deviance) Residuals*

Description

This function computes the Cox Model based on PLSR components computed model with

- as the response: the Survival time
- as explanatory variables: Xplan.

It uses the package `sgPLS` to perform group PLSR fit.

Usage

```
coxDKgplsDR(Xplan, ...)

## S3 method for class 'formula'
coxDKgplsDR(
  Xplan,
  time,
  time2,
  event,
  type,
  origin,
  typeres = "deviance",
  collapse,
  weighted,
  scaleX = TRUE,
  scaleY = TRUE,
  ncomp = min(7, ncol(Xplan)),
  modepls = "regression",
  ind.block.x,
  keepX,
  plot = FALSE,
  allres = FALSE,
  dataXplan = NULL,
  subset,
  weights,
  model_frame = FALSE,
  model_matrix = FALSE,
  contrasts.arg = NULL,
  kernel = "rbfdot",
  hyperkernel,
  verbose = FALSE,
  ...
)

## Default S3 method:
coxDKgplsDR(
  Xplan,
  time,
  time2,
  event,
  type,
  origin,
  typeres = "deviance",
  collapse,
  weighted,
  scaleX = TRUE,
  scaleY = TRUE,
  ncomp = min(7, ncol(Xplan)),
```

```
  modepls = "regression",
  ind.block.x,
  keepX,
  plot = FALSE,
  allres = FALSE,
  kernel = "rbfdot",
  hyperkernel,
  verbose = FALSE,
  ...
)
```

Arguments

Xplan	a formula or a matrix with the eXplanatory variables (training) dataset
...	Arguments to be passed on to survival::coxph.
time	for right censored data, this is the follow up time. For interval data, the first argument is the starting time for the interval.
time2	The status indicator, normally 0=alive, 1=dead. Other choices are TRUE/FALSE (TRUE = death) or 1/2 (2=death). For interval censored data, the status indicator is 0=right censored, 1=event at time, 2=left censored, 3=interval censored. Although unusual, the event indicator can be omitted, in which case all subjects are assumed to have an event.
event	ending time of the interval for interval censored or counting process data only. Intervals are assumed to be open on the left and closed on the right, (start, end]. For counting process data, event indicates whether an event occurred at the end of the interval.
type	character string specifying the type of censoring. Possible values are "right", "left", "counting", "interval", or "interval2". The default is "right" or "counting" depending on whether the time2 argument is absent or present, respectively.
origin	for counting process data, the hazard function origin. This option was intended to be used in conjunction with a model containing time dependent strata in order to align the subjects properly when they cross over from one strata to another, but it has rarely proven useful.
typeres	character string indicating the type of residual desired. Possible values are "martingale", "deviance", "score", "schoenfeld", "dfbeta", "dfbetas", and "scaledsch". Only enough of the string to determine a unique match is required.
collapse	vector indicating which rows to collapse (sum) over. In time-dependent models more than one row data can pertain to a single individual. If there were 4 individuals represented by 3, 1, 2 and 4 rows of data respectively, then collapse=c(1,1,1,2,3,3,4,4,4,4) could be used to obtain per subject rather than per observation residuals.
weighted	if TRUE and the model was fit with case weights, then the weighted residuals are returned.
scaleX	Should the Xplan columns be standardized ?
scaleY	Should the time values be standardized ?

ncomp — The number of components to include in the model. It this is not supplied, min(7,maximal number) components is used.

modepls — character string. What type of algorithm to use, (partially) matching one of "regression", "canonical". See gPLS for details

ind.block.x — a vector of integers describing the grouping of the X-variables. ind.block.x <- c(3,10,15) means that X is structured into 4 groups: X1 to X3; X4 to X10, X11 to X15 and X16 to Xp where p is the number of variables in the X matrix.

keepX — numeric vector of length ncomp, the number of variables to keep in X-loadings. By default all variables are kept in the model.

plot — Should the survival function be plotted ?)

allres — FALSE to return only the Cox model and TRUE for additionnal results. See details. Defaults to FALSE.

dataXplan — an optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If not found in dataXplan, the variables are taken from environment(Xplan), typically the environment from which coxpls is called.

subset — an optional vector specifying a subset of observations to be used in the fitting process.

weights — an optional vector of 'prior weights' to be used in the fitting process. Should be NULL or a numeric vector.

model_frame — If TRUE, the model frame is returned.

model_matrix — If TRUE, the model matrix is returned.

contrasts.arg — a list, whose entries are values (numeric matrices, functions or character strings naming functions) to be used as replacement values for the contrasts replacement function and whose names are the names of columns of data containing factors.

kernel — the kernel function used in training and predicting. This parameter can be set to any function, of class kernel, which computes the inner product in feature space between two vector arguments (see kernels). The kernlab package provides the most popular kernel functions which can be used by setting the kernel parameter to the following strings:

list("rbfdot") Radial Basis kernel "Gaussian"

list("polydot") Polynomial kernel

list("vanilladot") Linear kernel

list("tanhdot") Hyperbolic tangent kernel

list("laplacedot") Laplacian kernel

list("besseldot") Bessel kernel

list("anovadot") ANOVA RBF kernel

list("splinedot") Spline kernel

hyperkernel — the list of hyper-parameters (kernel parameters). This is a list which contains the parameters to be used with the kernel function. For valid parameters for existing kernels are :

- sigma, inverse kernel width for the Radial Basis kernel function "rbfdot" and the Laplacian kernel "laplacedot".

- `degree`, `scale`, `offset` for the Polynomial kernel "polydot".
- `scale`, offset for the Hyperbolic tangent kernel function "tanhdot".
- `sigma`, `order`, `degree` for the Bessel kernel "besseldot".
- `sigma`, `degree` for the ANOVA kernel "anovadot".

In the case of a Radial Basis kernel function (Gaussian) or Laplacian kernel, if `hyperkernel` is missing, the heuristics in sigest are used to calculate a good sigma value from the data.

`verbose` Should some details be displayed ?

Details

If `allres=FALSE` returns only the final Cox-model. If `allres=TRUE` returns a list with the PLS components, the final Cox-model and the group PLSR model. `allres=TRUE` is useful for evluating model prediction accuracy on a test sample.

Value

If `allres=FALSE` :

`cox_DKgplsDR` Final Cox-model.

If `allres=TRUE` :

`tt_DKgplsDR` PLSR components.

`cox_DKgplsDR` Final Cox-model.

`DKgplsDR_mod` The PLSR model.

Author(s)

Frédéric Bertrand
<frederic.bertrand@lecnam.net>
https://fbertran.github.io/homepage/

References

A group and Sparse Group Partial Least Square approach applied in Genomics context, Liquet Benoit, Lafaye de Micheaux, Boris Hejblum, Rodolphe Thiebaut (2016). Bioinformatics.

Deviance residuals-based sparse PLS and sparse kernel PLS regression for censored data, Philippe Bastien, Frederic Bertrand, Nicolas Meyer and Myriam Maumy-Bertrand (2015), Bioinformatics, 31(3):397-404, doi:10.1093/bioinformatics/btu660.

See Also

coxph, gPLS

Examples

```
data(micro.censure)
data(Xmicro.censure_compl_imp)

X_train_micro <- apply((as.matrix(Xmicro.censure_compl_imp)),FUN="as.numeric",MARGIN=2)[1:80,]
X_train_micro_df <- data.frame(X_train_micro)
Y_train_micro <- micro.censure$survyear[1:80]
C_train_micro <- micro.censure$DC[1:80]

(coxDKgplsDR_fit=coxDKgplsDR(X_train_micro,Y_train_micro,C_train_micro,
ncomp=6,ind.block.x=c(3,10,15),keepX=rep(4,6)))
(coxDKgplsDR_fit=coxDKgplsDR(~X_train_micro,Y_train_micro,C_train_micro,
ncomp=6,ind.block.x=c(3,10,15),keepX=rep(4,6)))
(coxDKgplsDR_fit=coxDKgplsDR(~.,Y_train_micro,C_train_micro,ncomp=6,
dataXplan=X_train_micro_df,ind.block.x=c(3,10,15),keepX=rep(4,6)))

rm(X_train_micro,Y_train_micro,C_train_micro,cox_spls_sgpls_fit)
```

`coxDKsgplsDR` *Fitting a Direct Kernel group sparse PLS model on the (Deviance) Residuals*

Description

This function computes the Cox Model based on PLSR components computed model with

- as the response: the Survival time
- as explanatory variables: Xplan.

It uses the package `sgplsDR` to perform group PLSR fit.

Usage

```
coxDKsgplsDR(Xplan, ...)

## S3 method for class 'formula'
coxDKsgplsDR(
  Xplan,
  time,
  time2,
  event,
  type,
  origin,
  typeres = "deviance",
  collapse,
  weighted,
  scaleX = TRUE,
```

```
  scaleY = TRUE,
  ncomp = min(7, ncol(Xplan)),
  modepls = "regression",
  ind.block.x,
  keepX,
  alpha.x,
  upper.lambda = 10^5,
  plot = FALSE,
  allres = FALSE,
  dataXplan = NULL,
  subset,
  weights,
  model_frame = FALSE,
  model_matrix = FALSE,
  contrasts.arg = NULL,
  kernel = "rbfdot",
  hyperkernel,
  verbose = FALSE,
  ...
)

## Default S3 method:
coxDKsgplsDR(
  Xplan,
  time,
  time2,
  event,
  type,
  origin,
  typeres = "deviance",
  collapse,
  weighted,
  scaleX = TRUE,
  scaleY = TRUE,
  ncomp = min(7, ncol(Xplan)),
  modepls = "regression",
  ind.block.x,
  keepX,
  alpha.x,
  upper.lambda = 10^5,
  plot = FALSE,
  allres = FALSE,
  kernel = "rbfdot",
  hyperkernel,
  verbose = FALSE,
  ...
)
```

Arguments

Xplan	a formula or a matrix with the eXplanatory variables (training) dataset
...	Arguments to be passed on to survival::coxph.
time	for right censored data, this is the follow up time. For interval data, the first argument is the starting time for the interval.
time2	The status indicator, normally 0=alive, 1=dead. Other choices are TRUE/FALSE (TRUE = death) or 1/2 (2=death). For interval censored data, the status indicator is 0=right censored, 1=event at time, 2=left censored, 3=interval censored. Although unusual, the event indicator can be omitted, in which case all subjects are assumed to have an event.
event	ending time of the interval for interval censored or counting process data only. Intervals are assumed to be open on the left and closed on the right, (start, end]. For counting process data, event indicates whether an event occurred at the end of the interval.
type	character string specifying the type of censoring. Possible values are "right", "left", "counting", "interval", or "interval2". The default is "right" or "counting" depending on whether the time2 argument is absent or present, respectively.
origin	for counting process data, the hazard function origin. This option was intended to be used in conjunction with a model containing time dependent strata in order to align the subjects properly when they cross over from one strata to another, but it has rarely proven useful.
typeres	character string indicating the type of residual desired. Possible values are "martingale", "deviance", "score", "schoenfeld", "dfbeta", "dfbetas", and "scaledsch". Only enough of the string to determine a unique match is required.
collapse	vector indicating which rows to collapse (sum) over. In time-dependent models more than one row data can pertain to a single individual. If there were 4 individuals represented by 3, 1, 2 and 4 rows of data respectively, then collapse=c(1,1,1,2,3,3,4,4,4,4) could be used to obtain per subject rather than per observation residuals.
weighted	if TRUE and the model was fit with case weights, then the weighted residuals are returned.
scaleX	Should the Xplan columns be standardized ?
scaleY	Should the time values be standardized ?
ncomp	The number of components to include in the model. It this is not supplied, min(7,maximal number) components is used.
modepls	character string. What type of algorithm to use, (partially) matching one of "regression", "canonical". See gPLS for details
ind.block.x	a vector of integers describing the grouping of the X-variables. ind.block.x <- c(3,10,15) means that X is structured into 4 groups: X1 to X3; X4 to X10, X11 to X15 and X16 to Xp where p is the number of variables in the X matrix.
keepX	numeric vector of length ncomp, the number of variables to keep in X-loadings. By default all variables are kept in the model.

`alpha.x` The mixing parameter (value between 0 and 1) related to the sparsity within group for the X dataset.

`upper.lambda` By default `upper.lambda=10^5`. A large value specifying the upper bound of the intervall of lambda values for searching the value of the tuning parameter (lambda) corresponding to a non-zero group of variables.

`plot` Should the survival function be plotted ?)

`allres` FALSE to return only the Cox model and TRUE for additionnal results. See details. Defaults to FALSE.

`dataXplan` an optional data frame, list or environment (or object coercible by `as.data.frame` to a data frame) containing the variables in the model. If not found in `dataXplan`, the variables are taken from `environment(Xplan)`, typically the environment from which `coxpls` is called.

`subset` an optional vector specifying a subset of observations to be used in the fitting process.

`weights` an optional vector of 'prior weights' to be used in the fitting process. Should be `NULL` or a numeric vector.

`model_frame` If `TRUE`, the model frame is returned.

`model_matrix` If `TRUE`, the model matrix is returned.

`contrasts.arg` a list, whose entries are values (numeric matrices, functions or character strings naming functions) to be used as replacement values for the contrasts replacement function and whose names are the names of columns of data containing factors.

`kernel` the kernel function used in training and predicting. This parameter can be set to any function, of class kernel, which computes the inner product in feature space between two vector arguments (see kernels). The `kernlab` package provides the most popular kernel functions which can be used by setting the kernel parameter to the following strings:

list("rbfdot") Radial Basis kernel "Gaussian"

list("polydot") Polynomial kernel

list("vanilladot") Linear kernel

list("tanhdot") Hyperbolic tangent kernel

list("laplacedot") Laplacian kernel

list("besseldot") Bessel kernel

list("anovadot") ANOVA RBF kernel

list("splinedot") Spline kernel

`hyperkernel` the list of hyper-parameters (kernel parameters). This is a list which contains the parameters to be used with the kernel function. For valid parameters for existing kernels are :

- `sigma`, inverse kernel width for the Radial Basis kernel function "rbfdot" and the Laplacian kernel "laplacedot".
- `degree`, `scale`, `offset` for the Polynomial kernel "polydot".
- `scale`, offset for the Hyperbolic tangent kernel function "tanhdot".
- `sigma`, `order`, `degree` for the Bessel kernel "besseldot".
- `sigma`, `degree` for the ANOVA kernel "anovadot".

	In the case of a Radial Basis kernel function (Gaussian) or Laplacian kernel, if hyperkernel is missing, the heuristics in sigest are used to calculate a good sigma value from the data.
verbose	Should some details be displayed ?

Details

If allres=FALSE returns only the final Cox-model. If allres=TRUE returns a list with the PLS components, the final Cox-model and the group PLSR model. allres=TRUE is useful for evluating model prediction accuracy on a test sample.

Value

If allres=FALSE :

cox_DKsgplsDR	Final Cox-model.

If allres=TRUE :

tt_DKsgplsDR	PLSR components.
cox_DKsgplsDR	Final Cox-model.
DKsgplsDR_mod	The PLSR model.

Author(s)

Frédéric Bertrand
<frederic.bertrand@lecnam.net>
https://fbertran.github.io/homepage/

References

A group and Sparse Group Partial Least Square approach applied in Genomics context, Liquet Benoit, Lafaye de Micheaux, Boris Hejblum, Rodolphe Thiebaut (2016). Bioinformatics.

Deviance residuals-based sparse PLS and sparse kernel PLS regression for censored data, Philippe Bastien, Frederic Bertrand, Nicolas Meyer and Myriam Maumy-Bertrand (2015), Bioinformatics, 31(3):397-404, doi:10.1093/bioinformatics/btu660.

See Also

coxph, gPLS

Examples

```
data(micro.censure)
data(Xmicro.censure_compl_imp)

X_train_micro <- apply((as.matrix(Xmicro.censure_compl_imp)),
FUN="as.numeric",MARGIN=2)[1:80,]
X_train_micro_df <- data.frame(X_train_micro)
```

```
Y_train_micro <- micro.censure$survyear[1:80]
C_train_micro <- micro.censure$DC[1:80]

(coxDKsgplsDR_fit=coxDKsgplsDR(X_train_micro,Y_train_micro,C_train_micro,
ncomp=6,ind.block.x=c(3,10,15), alpha.x = rep(0.95, 6)))
(coxDKsgplsDR_fit=coxDKsgplsDR(~X_train_micro,Y_train_micro,C_train_micro,
ncomp=6,ind.block.x=c(3,10,15), alpha.x = rep(0.95, 6)))
(coxDKsgplsDR_fit=coxDKsgplsDR(~.,Y_train_micro,C_train_micro,ncomp=6,
dataXplan=X_train_micro_df,ind.block.x=c(3,10,15), alpha.x = rep(0.95, 6)))

rm(X_train_micro,Y_train_micro,C_train_micro,coxDKsgplsDR_fit)
```

`coxDKspls_sgplsDR` *Fitting a Cox-Model on sparse PLSR components using the (Deviance) Residuals*

Description

This function computes the Cox Model based on PLSR components computed model with

- as the response: the Survival time
- as explanatory variables: Xplan.

It uses the package `sgPLS` to perform group PLSR fit.

Usage

```
coxDKspls_sgplsDR(Xplan, ...)

## S3 method for class 'formula'
coxDKspls_sgplsDR(
  Xplan,
  time,
  time2,
  event,
  type,
  origin,
  typeres = "deviance",
  collapse,
  weighted,
  scaleX = TRUE,
  scaleY = TRUE,
  ncomp = min(7, ncol(Xplan)),
  ind.block.x = NULL,
  modepls = "regression",
  keepX,
  plot = FALSE,
```

```
  allres = FALSE,
  dataXplan = NULL,
  subset,
  weights,
  model_frame = FALSE,
  model_matrix = FALSE,
  contrasts.arg = NULL,
  kernel = "rbfdot",
  hyperkernel,
  verbose = FALSE,
  ...
)

## Default S3 method:
coxDKspls_sgplsDR(
  Xplan,
  time,
  time2,
  event,
  type,
  origin,
  typeres = "deviance",
  collapse,
  weighted,
  scaleX = TRUE,
  scaleY = TRUE,
  ncomp = min(7, ncol(Xplan)),
  ind.block.x = NULL,
  modepls = "regression",
  keepX,
  alpha.x,
  upper.lambda = 10^5,
  plot = FALSE,
  allres = FALSE,
  kernel = "rbfdot",
  hyperkernel,
  verbose = FALSE,
  ...
)
```

Arguments

Xplan	a formula or a matrix with the eXplanatory variables (training) dataset
...	Arguments to be passed on to survival::coxph.
time	for right censored data, this is the follow up time. For interval data, the first argument is the starting time for the interval.
time2	The status indicator, normally 0=alive, 1=dead. Other choices are TRUE/FALSE (TRUE = death) or 1/2 (2=death). For interval censored data, the status indica-

tor is 0=right censored, 1=event at time, 2=left censored, 3=interval censored. Although unusual, the event indicator can be omitted, in which case all subjects are assumed to have an event.

event — ending time of the interval for interval censored or counting process data only. Intervals are assumed to be open on the left and closed on the right, (start, end]. For counting process data, event indicates whether an event occurred at the end of the interval.

type — character string specifying the type of censoring. Possible values are "right", "left", "counting", "interval", or "interval2". The default is "right" or "counting" depending on whether the time2 argument is absent or present, respectively.

origin — for counting process data, the hazard function origin. This option was intended to be used in conjunction with a model containing time dependent strata in order to align the subjects properly when they cross over from one strata to another, but it has rarely proven useful.

typeres — character string indicating the type of residual desired. Possible values are "martingale", "deviance", "score", "schoenfeld", "dfbeta", "dfbetas", and "scaledsch". Only enough of the string to determine a unique match is required.

collapse — vector indicating which rows to collapse (sum) over. In time-dependent models more than one row data can pertain to a single individual. If there were 4 individuals represented by 3, 1, 2 and 4 rows of data respectively, then collapse=c(1,1,1,2,3,3,4,4,4,4) could be used to obtain per subject rather than per observation residuals.

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

scaleX — Should the Xplan columns be standardized ?

scaleY — Should the time values be standardized ?

ncomp — The number of components to include in the model. It this is not supplied, min(7,maximal number) components is used.

ind.block.x — a vector of integers describing the grouping of the X-variables. ind.block.x <- c(3,10,15) means that X is structured into 4 groups: X1 to X3; X4 to X10, X11 to X15 and X16 to Xp where p is the number of variables in the X matrix.

modepls — character string. What type of algorithm to use, (partially) matching one of "regression", "canonical". See gPLS for details

keepX — numeric vector of length ncomp, the number of variables to keep in X-loadings. By default all variables are kept in the model.

plot — Should the survival function be plotted ?)

allres — FALSE to return only the Cox model and TRUE for additionnal results. See details. Defaults to FALSE.

dataXplan — an optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If not found in dataXplan, the variables are taken from environment(Xplan), typically the environment from which coxpls is called.

subset	an optional vector specifying a subset of observations to be used in the fitting process.
weights	an optional vector of 'prior weights' to be used in the fitting process. Should be NULL or a numeric vector.
model_frame	If TRUE, the model frame is returned.
model_matrix	If TRUE, the model matrix is returned.
contrasts.arg	a list, whose entries are values (numeric matrices, functions or character strings naming functions) to be used as replacement values for the contrasts replacement function and whose names are the names of columns of data containing factors.
kernel	the kernel function used in training and predicting. This parameter can be set to any function, of class kernel, which computes the inner product in feature space between two vector arguments (see kernels). The kernlab package provides the most popular kernel functions which can be used by setting the kernel parameter to the following strings: **list("rbfdot")** Radial Basis kernel "Gaussian" **list("polydot")** Polynomial kernel **list("vanilladot")** Linear kernel **list("tanhdot")** Hyperbolic tangent kernel **list("laplacedot")** Laplacian kernel **list("besseldot")** Bessel kernel **list("anovadot")** ANOVA RBF kernel **list("splinedot")** Spline kernel
hyperkernel	the list of hyper-parameters (kernel parameters). This is a list which contains the parameters to be used with the kernel function. For valid parameters for existing kernels are : • sigma, inverse kernel width for the Radial Basis kernel function "rbfdot" and the Laplacian kernel "laplacedot". • degree, scale, offset for the Polynomial kernel "polydot". • scale, offset for the Hyperbolic tangent kernel function "tanhdot". • sigma, order, degree for the Bessel kernel "besseldot". • sigma, degree for the ANOVA kernel "anovadot". In the case of a Radial Basis kernel function (Gaussian) or Laplacian kernel, if hyperkernel is missing, the heuristics in sigest are used to calculate a good sigma value from the data.
verbose	Should some details be displayed ?
alpha.x	numeric vector of length ncomp giving the sparsity level applied within each component. Required when ind.block.x is specified.
upper.lambda	numeric value controlling the maximal penalty considered by sgPLS when estimating sparse group loadings. Defaults to 10^5.

Details

If allres=FALSE returns only the final Cox-model. If allres=TRUE returns a list with the PLS components, the final Cox-model and the group PLSR model. allres=TRUE is useful for evluating model prediction accuracy on a test sample.

Value

If `allres=FALSE` :

`cox_DKspls_sgplsDR`
: Final Cox-model.

If `allres=TRUE` :

`tt_DKspls_sgplsDR`
: PLSR components.

`cox_DKspls_sgplsDR`
: Final Cox-model.

`DKspls_sgplsDR_mod`
: The PLSR model.

Author(s)

Frédéric Bertrand
<frederic.bertrand@lecnam.net>
https://fbertran.github.io/homepage/

References

A group and Sparse Group Partial Least Square approach applied in Genomics context, Liquet Benoit, Lafaye de Micheaux, Boris Hejblum, Rodolphe Thiebaut (2016). Bioinformatics.

Deviance residuals-based sparse PLS and sparse kernel PLS regression for censored data, Philippe Bastien, Frederic Bertrand, Nicolas Meyer and Myriam Maumy-Bertrand (2015), Bioinformatics, 31(3):397-404, doi:10.1093/bioinformatics/btu660.

See Also

coxph, gPLS

Examples

```
data(micro.censure)
data(Xmicro.censure_compl_imp)

X_train_micro <- apply((as.matrix(Xmicro.censure_compl_imp)),
FUN="as.numeric",MARGIN=2)[1:80,]
X_train_micro_df <- data.frame(X_train_micro)
Y_train_micro <- micro.censure$survyear[1:80]
C_train_micro <- micro.censure$DC[1:80]

(cox_DKspls_sgplsDR_fit=coxDKspls_sgplsDR(X_train_micro,Y_train_micro,
C_train_micro,ncomp=6,ind.block.x=c(3,10,15), alpha.x = rep(0.95, 6)))
(cox_DKspls_sgplsDR_fit=coxDKspls_sgplsDR(~X_train_micro,Y_train_micro,
C_train_micro,ncomp=6,ind.block.x=c(3,10,15), alpha.x = rep(0.95, 6)))
(cox_DKspls_sgplsDR_fit=coxDKspls_sgplsDR(~.,Y_train_micro,C_train_micro,
ncomp=6,dataXplan=X_train_micro_df,ind.block.x=c(3,10,15),
```

```
alpha.x = rep(0.95, 6)))

rm(X_train_micro,Y_train_micro,C_train_micro,cox_DKspls_sgplsDR_fit)
```

---

coxgpls *Fitting a Cox-Model on group PLSR components*

Description

This function computes the Cox Model based on PLSR components computed model with

- as the response: the Survival time
- as explanatory variables: Xplan.

It uses the package sgPLS to perform group PLSR fit.

Usage

```
coxgpls(Xplan, ...)

## S3 method for class 'formula'
coxgpls(
  Xplan,
  time,
  time2,
  event,
  type,
  origin,
  typeres = "deviance",
  collapse,
  weighted,
  scaleX = TRUE,
  scaleY = TRUE,
  ncomp = min(7, ncol(Xplan)),
  modepls = "regression",
  ind.block.x,
  keepX,
  plot = FALSE,
  allres = FALSE,
  dataXplan = NULL,
  subset,
  weights,
  model_frame = FALSE,
  model_matrix = FALSE,
  contrasts.arg = NULL,
  ...
)
```

```
## Default S3 method:
coxgpls(
  Xplan,
  time,
  time2,
  event,
  type,
  origin,
  typeres = "deviance",
  collapse,
  weighted,
  scaleX = TRUE,
  scaleY = TRUE,
  ncomp = min(7, ncol(Xplan)),
  modepls = "regression",
  ind.block.x,
  keepX,
  plot = FALSE,
  allres = FALSE,
  ...
)
```

Arguments

Xplan	a formula or a matrix with the eXplanatory variables (training) dataset
...	Arguments to be passed on to survival::coxph.
time	for right censored data, this is the follow up time. For interval data, the first argument is the starting time for the interval.
time2	The status indicator, normally 0=alive, 1=dead. Other choices are TRUE/FALSE (TRUE = death) or 1/2 (2=death). For interval censored data, the status indicator is 0=right censored, 1=event at time, 2=left censored, 3=interval censored. Although unusual, the event indicator can be omitted, in which case all subjects are assumed to have an event.
event	ending time of the interval for interval censored or counting process data only. Intervals are assumed to be open on the left and closed on the right, (start, end]. For counting process data, event indicates whether an event occurred at the end of the interval.
type	character string specifying the type of censoring. Possible values are "right", "left", "counting", "interval", or "interval2". The default is "right" or "counting" depending on whether the time2 argument is absent or present, respectively.
origin	for counting process data, the hazard function origin. This option was intended to be used in conjunction with a model containing time dependent strata in order to align the subjects properly when they cross over from one strata to another, but it has rarely proven useful.

typeres	character string indicating the type of residual desired. Possible values are "martingale", "deviance", "score", "schoenfeld", "dfbeta", "dfbetas", and "scaledsch". Only enough of the string to determine a unique match is required.
collapse	vector indicating which rows to collapse (sum) over. In time-dependent models more than one row data can pertain to a single individual. If there were 4 individuals represented by 3, 1, 2 and 4 rows of data respectively, then collapse=c(1,1,1,2,3,3,4,4,4,4) could be used to obtain per subject rather than per observation residuals.
weighted	if TRUE and the model was fit with case weights, then the weighted residuals are returned.
scaleX	Should the Xplan columns be standardized ?
scaleY	Should the time values be standardized ?
ncomp	The number of components to include in the model. It this is not supplied, min(7,maximal number) components is used.
modepls	character string. What type of algorithm to use, (partially) matching one of "regression", "canonical". See gPLS for details
ind.block.x	a vector of integers describing the grouping of the X-variables. ind.block.x <- c(3,10,15) means that X is structured into 4 groups: X1 to X3; X4 to X10, X11 to X15 and X16 to Xp where p is the number of variables in the X matrix. When missing, every predictor is placed in its own group.
keepX	numeric vector of length ncomp, the number of variables to keep in X-loadings. By default all variables are kept in the model.
plot	Should the survival function be plotted ?)
allres	FALSE to return only the Cox model and TRUE for additionnal results. See details. Defaults to FALSE.
dataXplan	an optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If not found in dataXplan, the variables are taken from environment(Xplan), typically the environment from which coxpls is called.
subset	an optional vector specifying a subset of observations to be used in the fitting process.
weights	an optional vector of 'prior weights' to be used in the fitting process. Should be NULL or a numeric vector.
model_frame	If TRUE, the model frame is returned.
model_matrix	If TRUE, the model matrix is returned.
contrasts.arg	a list, whose entries are values (numeric matrices, functions or character strings naming functions) to be used as replacement values for the contrasts replacement function and whose names are the names of columns of data containing factors.

Details

If allres=FALSE returns only the final Cox-model. If allres=TRUE returns a list with the PLS components, the final Cox-model and the group PLSR model. allres=TRUE is useful for evluating model prediction accuracy on a test sample.

Value

If `allres=FALSE` :

`cox_gpls`	Final Cox-model.

If `allres=TRUE` :

`tt_gpls`	PLSR components.
`cox_gpls`	Final Cox-model.
`gpls_mod`	The PLSR model.

Author(s)

Frédéric Bertrand
<frederic.bertrand@lecnam.net>
https://fbertran.github.io/homepage/

References

A group and Sparse Group Partial Least Square approach applied in Genomics context, Liquet Benoit, Lafaye de Micheaux, Boris Hejblum, Rodolphe Thiebaut (2016). Bioinformatics.

Deviance residuals-based sparse PLS and sparse kernel PLS regression for censored data, Philippe Bastien, Frederic Bertrand, Nicolas Meyer and Myriam Maumy-Bertrand (2015), Bioinformatics, 31(3):397-404, doi:10.1093/bioinformatics/btu660.

See Also

coxph, gPLS

Examples

```
data(micro.censure)
data(Xmicro.censure_compl_imp)

X_train_micro <- apply((as.matrix(Xmicro.censure_compl_imp)),FUN="as.numeric",MARGIN=2)[1:80,]
X_train_micro_df <- data.frame(X_train_micro)
Y_train_micro <- micro.censure$survyear[1:80]
C_train_micro <- micro.censure$DC[1:80]

(coxgpls_fit=coxgpls(X_train_micro,Y_train_micro,C_train_micro,
ncomp=6,keepX=rep(4,6)))
(coxgpls_fit=coxgpls(~X_train_micro,Y_train_micro,C_train_micro,
ncomp=6,keepX=rep(4,6)))
(ccoxgpls_fit=coxgpls(~.,Y_train_micro,C_train_micro,ncomp=6,
dataXplan=X_train_micro_df,keepX=rep(4,6)))

rm(X_train_micro,Y_train_micro,C_train_micro,cox_spls_sgpls_fit)
```

coxgplsDR *Fitting a Cox-Model on group PLSR components using the (Deviance) Residuals*

Description

This function computes the Cox Model based on PLSR components computed model with

- as the response: the Survival time
- as explanatory variables: Xplan.

It uses the package sgPLS to perform group PLSR fit.

Usage

```
coxgplsDR(Xplan, ...)

## S3 method for class 'formula'
coxgplsDR(
  Xplan,
  time,
  time2,
  event,
  type,
  origin,
  typeres = "deviance",
  collapse,
  weighted,
  scaleX = TRUE,
  scaleY = TRUE,
  ncomp = min(7, ncol(Xplan)),
  modepls = "regression",
  ind.block.x,
  keepX,
  plot = FALSE,
  allres = FALSE,
  dataXplan = NULL,
  subset,
  weights,
  model_frame = FALSE,
  model_matrix = FALSE,
  contrasts.arg = NULL,
  ...
)

## Default S3 method:
coxgplsDR(
```

```
  Xplan,
  time,
  time2,
  event,
  type,
  origin,
  typeres = "deviance",
  collapse,
  weighted,
  scaleX = TRUE,
  scaleY = TRUE,
  ncomp = min(7, ncol(Xplan)),
  modepls = "regression",
  ind.block.x,
  keepX,
  plot = FALSE,
  allres = FALSE,
  ...
)
```

Arguments

Xplan	a formula or a matrix with the eXplanatory variables (training) dataset
...	Arguments to be passed on to survival::coxph.
time	for right censored data, this is the follow up time. For interval data, the first argument is the starting time for the interval.
time2	The status indicator, normally 0=alive, 1=dead. Other choices are TRUE/FALSE (TRUE = death) or 1/2 (2=death). For interval censored data, the status indicator is 0=right censored, 1=event at time, 2=left censored, 3=interval censored. Although unusual, the event indicator can be omitted, in which case all subjects are assumed to have an event.
event	ending time of the interval for interval censored or counting process data only. Intervals are assumed to be open on the left and closed on the right, (start, end]. For counting process data, event indicates whether an event occurred at the end of the interval.
type	character string specifying the type of censoring. Possible values are "right", "left", "counting", "interval", or "interval2". The default is "right" or "counting" depending on whether the time2 argument is absent or present, respectively.
origin	for counting process data, the hazard function origin. This option was intended to be used in conjunction with a model containing time dependent strata in order to align the subjects properly when they cross over from one strata to another, but it has rarely proven useful.
typeres	character string indicating the type of residual desired. Possible values are "martingale", "deviance", "score", "schoenfeld", "dfbeta", "dfbetas", and "scaledsch". Only enough of the string to determine a unique match is required.

collapse	vector indicating which rows to collapse (sum) over. In time-dependent models more than one row data can pertain to a single individual. If there were 4 individuals represented by 3, 1, 2 and 4 rows of data respectively, then collapse=c(1,1,1,2,3,3,4,4,4,4) could be used to obtain per subject rather than per observation residuals.
weighted	if TRUE and the model was fit with case weights, then the weighted residuals are returned.
scaleX	Should the Xplan columns be standardized ?
scaleY	Should the time values be standardized ?
ncomp	The number of components to include in the model. It this is not supplied, min(7,maximal number) components is used.
modepls	character string. What type of algorithm to use, (partially) matching one of "regression", "canonical". See gPLS for details
ind.block.x	a vector of integers describing the grouping of the X-variables. ind.block.x <- c(3,10,15) means that X is structured into 4 groups: X1 to X3; X4 to X10, X11 to X15 and X16 to Xp where p is the number of variables in the X matrix.
keepX	numeric vector of length ncomp, the number of variables to keep in X-loadings. By default all variables are kept in the model.
plot	Should the survival function be plotted ?)
allres	FALSE to return only the Cox model and TRUE for additionnal results. See details. Defaults to FALSE.
dataXplan	an optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If not found in dataXplan, the variables are taken from environment(Xplan), typically the environment from which coxpls is called.
subset	an optional vector specifying a subset of observations to be used in the fitting process.
weights	an optional vector of 'prior weights' to be used in the fitting process. Should be NULL or a numeric vector.
model_frame	If TRUE, the model frame is returned.
model_matrix	If TRUE, the model matrix is returned.
contrasts.arg	a list, whose entries are values (numeric matrices, functions or character strings naming functions) to be used as replacement values for the contrasts replacement function and whose names are the names of columns of data containing factors.

Details

If allres=FALSE returns only the final Cox-model. If allres=TRUE returns a list with the PLS components, the final Cox-model and the group PLSR model. allres=TRUE is useful for evluating model prediction accuracy on a test sample.

Value

If allres=FALSE :

cox_gplsDR	Final Cox-model.

If `allres=TRUE` :

`tt_gplsDR`	PLSR components.
`cox_gplsDR`	Final Cox-model.
`gplsDR_mod`	The PLSR model.

Author(s)

Frédéric Bertrand
<frederic.bertrand@lecnam.net>
https://fbertran.github.io/homepage/

References

A group and Sparse Group Partial Least Square approach applied in Genomics context, Liquet Benoit, Lafaye de Micheaux, Boris Hejblum, Rodolphe Thiebaut (2016). Bioinformatics.

Deviance residuals-based sparse PLS and sparse kernel PLS regression for censored data, Philippe Bastien, Frederic Bertrand, Nicolas Meyer and Myriam Maumy-Bertrand (2015), Bioinformatics, 31(3):397-404, doi:10.1093/bioinformatics/btu660.

See Also

coxph, gPLS

Examples

```
data(micro.censure)
data(Xmicro.censure_compl_imp)

X_train_micro <- apply((as.matrix(Xmicro.censure_compl_imp)),FUN="as.numeric",MARGIN=2)[1:80,]
X_train_micro_df <- data.frame(X_train_micro)
Y_train_micro <- micro.censure$survyear[1:80]
C_train_micro <- micro.censure$DC[1:80]

(coxgplsDR_fit=coxgplsDR(X_train_micro,Y_train_micro,C_train_micro,
ncomp=6,ind.block.x=c(3,10,15),keepX=rep(4,6)))
(coxgplsDR_fit=coxgplsDR(~X_train_micro,Y_train_micro,C_train_micro,
ncomp=6,ind.block.x=c(3,10,15),keepX=rep(4,6)))
(coxgplsDR_fit=coxgplsDR(~.,Y_train_micro,C_train_micro,ncomp=6,
dataXplan=X_train_micro_df,ind.block.x=c(3,10,15),keepX=rep(4,6)))

rm(X_train_micro,Y_train_micro,C_train_micro,cox_spls_sgpls_fit)
```

coxsgpls *Fitting a Cox-Model on group sparse PLSR components*

Description

This function computes the Cox Model based on PLSR components computed model with

- as the response: the Survival time
- as explanatory variables: Xplan.

It uses the package sgPLS to perform group PLSR fit.

Usage

```
coxsgpls(Xplan, ...)

## S3 method for class 'formula'
coxsgpls(
  Xplan,
  time,
  time2,
  event,
  type,
  origin,
  typeres = "deviance",
  collapse,
  weighted,
  scaleX = TRUE,
  scaleY = TRUE,
  ncomp = min(7, ncol(Xplan)),
  modepls = "regression",
  ind.block.x,
  keepX,
  alpha.x,
  upper.lambda = 10^5,
  plot = FALSE,
  allres = FALSE,
  dataXplan = NULL,
  subset,
  weights,
  model_frame = FALSE,
  model_matrix = FALSE,
  contrasts.arg = NULL,
  ...
)

## Default S3 method:
```

```
coxsgpls(
  Xplan,
  time,
  time2,
  event,
  type,
  origin,
  typeres = "deviance",
  collapse,
  weighted,
  scaleX = TRUE,
  scaleY = TRUE,
  ncomp = min(7, ncol(Xplan)),
  modepls = "regression",
  ind.block.x,
  keepX,
  alpha.x,
  upper.lambda = 10^5,
  plot = FALSE,
  allres = FALSE,
  ...
)
```

Arguments

Xplan	a formula or a matrix with the eXplanatory variables (training) dataset
...	Arguments to be passed on to survival::coxph.
time	for right censored data, this is the follow up time. For interval data, the first argument is the starting time for the interval.
time2	The status indicator, normally 0=alive, 1=dead. Other choices are TRUE/FALSE (TRUE = death) or 1/2 (2=death). For interval censored data, the status indicator is 0=right censored, 1=event at time, 2=left censored, 3=interval censored. Although unusual, the event indicator can be omitted, in which case all subjects are assumed to have an event.
event	ending time of the interval for interval censored or counting process data only. Intervals are assumed to be open on the left and closed on the right, (start, end]. For counting process data, event indicates whether an event occurred at the end of the interval.
type	character string specifying the type of censoring. Possible values are "right", "left", "counting", "interval", or "interval2". The default is "right" or "counting" depending on whether the time2 argument is absent or present, respectively.
origin	for counting process data, the hazard function origin. This option was intended to be used in conjunction with a model containing time dependent strata in order to align the subjects properly when they cross over from one strata to another, but it has rarely proven useful.

typeres	character string indicating the type of residual desired. Possible values are "martingale", "deviance", "score", "schoenfeld", "dfbeta", "dfbetas", and "scaledsch". Only enough of the string to determine a unique match is required.
collapse	vector indicating which rows to collapse (sum) over. In time-dependent models more than one row data can pertain to a single individual. If there were 4 individuals represented by 3, 1, 2 and 4 rows of data respectively, then collapse=c(1,1,1,2,3,3,4,4,4,4) could be used to obtain per subject rather than per observation residuals.
weighted	if TRUE and the model was fit with case weights, then the weighted residuals are returned.
scaleX	Should the Xplan columns be standardized ?
scaleY	Should the time values be standardized ?
ncomp	The number of components to include in the model. It this is not supplied, min(7,maximal number) components is used.
modepls	character string. What type of algorithm to use, (partially) matching one of "regression", "canonical". See gPLS for details
ind.block.x	a vector of integers describing the grouping of the X-variables. ind.block.x <- c(3,10,15) means that X is structured into 4 groups: X1 to X3; X4 to X10, X11 to X15 and X16 to Xp where p is the number of variables in the X matrix.
keepX	numeric vector of length ncomp, the number of variables to keep in X-loadings. By default all variables are kept in the model.
alpha.x	The mixing parameter (value between 0 and 1) related to the sparsity within group for the X dataset.
upper.lambda	By default upper.lambda=10^5. A large value specifying the upper bound of the intervall of lambda values for searching the value of the tuning parameter (lambda) corresponding to a non-zero group of variables.
plot	Should the survival function be plotted ?)
allres	FALSE to return only the Cox model and TRUE for additionnal results. See details. Defaults to FALSE.
dataXplan	an optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If not found in dataXplan, the variables are taken from environment(Xplan), typically the environment from which coxpls is called.
subset	an optional vector specifying a subset of observations to be used in the fitting process.
weights	an optional vector of 'prior weights' to be used in the fitting process. Should be NULL or a numeric vector.
model_frame	If TRUE, the model frame is returned.
model_matrix	If TRUE, the model matrix is returned.
contrasts.arg	a list, whose entries are values (numeric matrices, functions or character strings naming functions) to be used as replacement values for the contrasts replacement function and whose names are the names of columns of data containing factors.

Details

If `allres=FALSE` returns only the final Cox-model. If `allres=TRUE` returns a list with the PLS components, the final Cox-model and the group PLSR model. `allres=TRUE` is useful for evluating model prediction accuracy on a test sample.

Value

If `allres=FALSE` :

`cox_sgpls`	Final Cox-model.

If `allres=TRUE` :

`tt_sgpls`	PLSR components.
`cox_sgpls`	Final Cox-model.
`sgpls_mod`	The PLSR model.

Author(s)

Frédéric Bertrand
<frederic.bertrand@lecnam.net>
https://fbertran.github.io/homepage/

References

A group and Sparse Group Partial Least Square approach applied in Genomics context, Liquet Benoit, Lafaye de Micheaux, Boris Hejblum, Rodolphe Thiebaut (2016). Bioinformatics.

Deviance residuals-based sparse PLS and sparse kernel PLS regression for censored data, Philippe Bastien, Frederic Bertrand, Nicolas Meyer and Myriam Maumy-Bertrand (2015), Bioinformatics, 31(3):397-404, doi:10.1093/bioinformatics/btu660.

See Also

coxph, gPLS

Examples

```
data(micro.censure)
data(Xmicro.censure_compl_imp)

X_train_micro <- apply((as.matrix(Xmicro.censure_compl_imp)),FUN="as.numeric",MARGIN=2)[1:80,]
X_train_micro_df <- data.frame(X_train_micro)
Y_train_micro <- micro.censure$survyear[1:80]
C_train_micro <- micro.censure$DC[1:80]

(coxsgpls_fit=coxsgpls(X_train_micro,Y_train_micro,C_train_micro,
ncomp=6,ind.block.x=c(3,10,15), alpha.x = rep(0.95, 6)))
(coxsgpls_fit=coxsgpls(~X_train_micro,Y_train_micro,C_train_micro,
ncomp=6,ind.block.x=c(3,10,15), alpha.x = rep(0.95, 6)))
```

```
(coxsgpls_fit=coxsgpls(~.,Y_train_micro,C_train_micro,ncomp=6,
dataXplan=X_train_micro_df,ind.block.x=c(3,10,15), alpha.x = rep(0.95, 6)))

rm(X_train_micro,Y_train_micro,C_train_micro,cox_sgpls_sgfit)
```

`coxsgplsDR` *Fitting a Cox-Model on group sparse PLSR components using the (Deviance) Residuals*

Description

This function computes the Cox Model based on PLSR components computed model with

- as the response: the Survival time
- as explanatory variables: Xplan.

It uses the package `sgplsDR` to perform group PLSR fit.

Usage

```
coxsgplsDR(Xplan, ...)

## S3 method for class 'formula'
coxsgplsDR(
  Xplan,
  time,
  time2,
  event,
  type,
  origin,
  typeres = "deviance",
  collapse,
  weighted,
  scaleX = TRUE,
  scaleY = TRUE,
  ncomp = min(7, ncol(Xplan)),
  modepls = "regression",
  ind.block.x,
  keepX,
  alpha.x,
  upper.lambda = 10^5,
  plot = FALSE,
  allres = FALSE,
  dataXplan = NULL,
  subset,
  weights,
  model_frame = FALSE,
```

```
  model_matrix = FALSE,
  contrasts.arg = NULL,
  ...
)

## Default S3 method:
coxsgplsDR(
  Xplan,
  time,
  time2,
  event,
  type,
  origin,
  typeres = "deviance",
  collapse,
  weighted,
  scaleX = TRUE,
  scaleY = TRUE,
  ncomp = min(7, ncol(Xplan)),
  modepls = "regression",
  ind.block.x,
  keepX,
  alpha.x,
  upper.lambda = 10^5,
  plot = FALSE,
  allres = FALSE,
  ...
)
```

Arguments

Xplan	a formula or a matrix with the eXplanatory variables (training) dataset
...	Arguments to be passed on to survival::coxph.
time	for right censored data, this is the follow up time. For interval data, the first argument is the starting time for the interval.
time2	The status indicator, normally 0=alive, 1=dead. Other choices are TRUE/FALSE (TRUE = death) or 1/2 (2=death). For interval censored data, the status indicator is 0=right censored, 1=event at time, 2=left censored, 3=interval censored. Although unusual, the event indicator can be omitted, in which case all subjects are assumed to have an event.
event	ending time of the interval for interval censored or counting process data only. Intervals are assumed to be open on the left and closed on the right, (start, end]. For counting process data, event indicates whether an event occurred at the end of the interval.
type	character string specifying the type of censoring. Possible values are "right", "left", "counting", "interval", or "interval2". The default is "right" or "counting" depending on whether the time2 argument is absent or present, respectively.

origin	for counting process data, the hazard function origin. This option was intended to be used in conjunction with a model containing time dependent strata in order to align the subjects properly when they cross over from one strata to another, but it has rarely proven useful.
typeres	character string indicating the type of residual desired. Possible values are "martingale", "deviance", "score", "schoenfeld", "dfbeta", "dfbetas", and "scaledsch". Only enough of the string to determine a unique match is required.
collapse	vector indicating which rows to collapse (sum) over. In time-dependent models more than one row data can pertain to a single individual. If there were 4 individuals represented by 3, 1, 2 and 4 rows of data respectively, then collapse=c(1,1,1,2,3,3,4,4,4,4) could be used to obtain per subject rather than per observation residuals.
weighted	if TRUE and the model was fit with case weights, then the weighted residuals are returned.
scaleX	Should the Xplan columns be standardized ?
scaleY	Should the time values be standardized ?
ncomp	The number of components to include in the model. It this is not supplied, min(7,maximal number) components is used.
modepls	character string. What type of algorithm to use, (partially) matching one of "regression", "canonical". See gPLS for details
ind.block.x	a vector of integers describing the grouping of the X-variables. ind.block.x <- c(3,10,15) means that X is structured into 4 groups: X1 to X3; X4 to X10, X11 to X15 and X16 to Xp where p is the number of variables in the X matrix.
keepX	numeric vector of length ncomp, the number of variables to keep in X-loadings. By default all variables are kept in the model.
alpha.x	The mixing parameter (value between 0 and 1) related to the sparsity within group for the X dataset.
upper.lambda	By default upper.lambda=10^5. A large value specifying the upper bound of the intervall of lambda values for searching the value of the tuning parameter (lambda) corresponding to a non-zero group of variables.
plot	Should the survival function be plotted ?)
allres	FALSE to return only the Cox model and TRUE for additionnal results. See details. Defaults to FALSE.
dataXplan	an optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If not found in dataXplan, the variables are taken from environment(Xplan), typically the environment from which coxpls is called.
subset	an optional vector specifying a subset of observations to be used in the fitting process.
weights	an optional vector of 'prior weights' to be used in the fitting process. Should be NULL or a numeric vector.
model_frame	If TRUE, the model frame is returned.
model_matrix	If TRUE, the model matrix is returned.

`contrasts.arg` a list, whose entries are values (numeric matrices, functions or character strings naming functions) to be used as replacement values for the contrasts replacement function and whose names are the names of columns of data containing factors.

Details

If `allres=FALSE` returns only the final Cox-model. If `allres=TRUE` returns a list with the PLS components, the final Cox-model and the group PLSR model. `allres=TRUE` is useful for evluating model prediction accuracy on a test sample.

Value

If `allres=FALSE` :

`cox_sgplsDR` Final Cox-model.

If `allres=TRUE` :

`tt_sgplsDR` PLSR components.

`cox_sgplsDR` Final Cox-model.

`sgplsDR_mod` The PLSR model.

Author(s)

Frédéric Bertrand
<frederic.bertrand@lecnam.net>
https://fbertran.github.io/homepage/

References

A group and Sparse Group Partial Least Square approach applied in Genomics context, Liquet Benoit, Lafaye de Micheaux, Boris Hejblum, Rodolphe Thiebaut (2016). Bioinformatics.

Deviance residuals-based sparse PLS and sparse kernel PLS regression for censored data, Philippe Bastien, Frederic Bertrand, Nicolas Meyer and Myriam Maumy-Bertrand (2015), Bioinformatics, 31(3):397-404, doi:10.1093/bioinformatics/btu660.

See Also

coxph, gPLS

Examples

```
data(micro.censure)
data(Xmicro.censure_compl_imp)

X_train_micro <- apply((as.matrix(Xmicro.censure_compl_imp)),FUN="as.numeric",MARGIN=2)[1:80,]
X_train_micro_df <- data.frame(X_train_micro)
Y_train_micro <- micro.censure$survyear[1:80]
C_train_micro <- micro.censure$DC[1:80]
```

```
(coxsgplsDR_fit=coxsgplsDR(X_train_micro,Y_train_micro,C_train_micro,
ncomp=6,ind.block.x=c(3,10,15), alpha.x = rep(0.95, 6)))
(coxsgplsDR_fit=coxsgplsDR(~X_train_micro,Y_train_micro,C_train_micro,
ncomp=6,ind.block.x=c(3,10,15), alpha.x = rep(0.95, 6)))
(coxsgplsDR_fit=coxsgplsDR(~.,Y_train_micro,C_train_micro,ncomp=6,
dataXplan=X_train_micro_df,ind.block.x=c(3,10,15), alpha.x = rep(0.95, 6)))

rm(X_train_micro,Y_train_micro,C_train_micro,cox_sgplsDR_sgfit)
```

`coxspls_sgpls` *Fitting a Cox-Model on sparse PLSR components*

Description

This function computes the Cox Model based on PLSR components computed model with

- as the response: the Survival time
- as explanatory variables: Xplan.

It uses the package `sgPLS` to perform group PLSR fit.

Usage

```
coxspls_sgpls(Xplan, ...)

## S3 method for class 'formula'
coxspls_sgpls(
  Xplan,
  time,
  time2,
  event,
  type,
  origin,
  typeres = "deviance",
  collapse,
  weighted,
  scaleX = TRUE,
  scaleY = TRUE,
  ncomp = min(7, ncol(Xplan)),
  ind.block.x = NULL,
  modepls = "regression",
  keepX,
  plot = FALSE,
  allres = FALSE,
  dataXplan = NULL,
  subset,
  weights,
```

```
  model_frame = FALSE,
  model_matrix = FALSE,
  contrasts.arg = NULL,
  ...
)

## Default S3 method:
coxspls_sgpls(
  Xplan,
  time,
  time2,
  event,
  type,
  origin,
  typeres = "deviance",
  collapse,
  weighted,
  scaleX = TRUE,
  scaleY = TRUE,
  ncomp = min(7, ncol(Xplan)),
  ind.block.x = NULL,
  modepls = "regression",
  keepX,
  alpha.x,
  upper.lambda = 10^5,
  plot = FALSE,
  allres = FALSE,
  ...
)
```

Arguments

Xplan	a formula or a matrix with the eXplanatory variables (training) dataset
...	Arguments to be passed on to survival::coxph.
time	for right censored data, this is the follow up time. For interval data, the first argument is the starting time for the interval.
time2	The status indicator, normally 0=alive, 1=dead. Other choices are TRUE/FALSE (TRUE = death) or 1/2 (2=death). For interval censored data, the status indicator is 0=right censored, 1=event at time, 2=left censored, 3=interval censored. Although unusual, the event indicator can be omitted, in which case all subjects are assumed to have an event.
event	ending time of the interval for interval censored or counting process data only. Intervals are assumed to be open on the left and closed on the right, (start, end]. For counting process data, event indicates whether an event occurred at the end of the interval.
type	character string specifying the type of censoring. Possible values are "right", "left", "counting", "interval", or "interval2". The default is "right"

	or "counting" depending on whether the time2 argument is absent or present, respectively.
origin	for counting process data, the hazard function origin. This option was intended to be used in conjunction with a model containing time dependent strata in order to align the subjects properly when they cross over from one strata to another, but it has rarely proven useful.
typeres	character string indicating the type of residual desired. Possible values are "martingale", "deviance", "score", "schoenfeld", "dfbeta", "dfbetas", and "scaledsch". Only enough of the string to determine a unique match is required.
collapse	vector indicating which rows to collapse (sum) over. In time-dependent models more than one row data can pertain to a single individual. If there were 4 individuals represented by 3, 1, 2 and 4 rows of data respectively, then collapse=c(1,1,1,2,3,3,4,4,4,4) could be used to obtain per subject rather than per observation residuals.
weighted	if TRUE and the model was fit with case weights, then the weighted residuals are returned.
scaleX	Should the Xplan columns be standardized ?
scaleY	Should the time values be standardized ?
ncomp	The number of components to include in the model. It this is not supplied, min(7,maximal number) components is used.
ind.block.x	a vector of integers describing the grouping of the X-variables. ind.block.x <- c(3,10,15) means that X is structured into 4 groups: X1 to X3; X4 to X10, X11 to X15 and X16 to Xp where p is the number of variables in the X matrix.
modepls	character string. What type of algorithm to use, (partially) matching one of "regression", "canonical". See gPLS for details
keepX	numeric vector of length ncomp, the number of variables to keep in X-loadings. By default all variables are kept in the model.
plot	Should the survival function be plotted ?)
allres	FALSE to return only the Cox model and TRUE for additionnal results. See details. Defaults to FALSE.
dataXplan	an optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If not found in dataXplan, the variables are taken from environment(Xplan), typically the environment from which coxpls is called.
subset	an optional vector specifying a subset of observations to be used in the fitting process.
weights	an optional vector of 'prior weights' to be used in the fitting process. Should be NULL or a numeric vector.
model_frame	If TRUE, the model frame is returned.
model_matrix	If TRUE, the model matrix is returned.
contrasts.arg	a list, whose entries are values (numeric matrices, functions or character strings naming functions) to be used as replacement values for the contrasts replacement function and whose names are the names of columns of data containing factors.

alpha.x	numeric vector of length ncomp giving the sparsity level applied within each component. Required when ind.block.x is specified.
upper.lambda	numeric value controlling the maximal penalty considered by sgPLS when estimating sparse group loadings. Defaults to 10^5.

Details

If allres=FALSE returns only the final Cox-model. If allres=TRUE returns a list with the PLS components, the final Cox-model and the group PLSR model. allres=TRUE is useful for evluating model prediction accuracy on a test sample.

Value

If allres=FALSE :

cox_spls_sgpls	Final Cox-model.

If allres=TRUE :

tt_spls_sgpls	PLSR components.
cox_spls_sgpls	Final Cox-model.
spls_sgpls_mod	The PLSR model.

Author(s)

Frédéric Bertrand
<frederic.bertrand@lecnam.net>
https://fbertran.github.io/homepage/

References

A group and Sparse Group Partial Least Square approach applied in Genomics context, Liquet Benoit, Lafaye de Micheaux, Boris Hejblum, Rodolphe Thiebaut (2016). Bioinformatics.

Deviance residuals-based sparse PLS and sparse kernel PLS regression for censored data, Philippe Bastien, Frederic Bertrand, Nicolas Meyer and Myriam Maumy-Bertrand (2015), Bioinformatics, 31(3):397-404, doi:10.1093/bioinformatics/btu660.

See Also

coxph, gPLS

Examples

```
data(micro.censure)
data(Xmicro.censure_compl_imp)

X_train_micro <- apply((as.matrix(Xmicro.censure_compl_imp)),
FUN="as.numeric",MARGIN=2)[1:80,]
X_train_micro_df <- data.frame(X_train_micro)
```

```
Y_train_micro <- micro.censure$survyear[1:80]
C_train_micro <- micro.censure$DC[1:80]

(cox_spls_sgpls_fit=coxspls_sgpls(X_train_micro,Y_train_micro,C_train_micro,
ncomp=6,ind.block.x=c(3,10,15), alpha.x = rep(0.95, 6)))
(cox_spls_sgpls_fit=coxspls_sgpls(~X_train_micro,Y_train_micro,C_train_micro,
ncomp=6,ind.block.x=c(3,10,15), alpha.x = rep(0.95, 6)))
(cox_spls_sgpls_fit=coxspls_sgpls(~.,Y_train_micro,C_train_micro,ncomp=6,
dataXplan=X_train_micro_df,ind.block.x=c(3,10,15), alpha.x = rep(0.95, 6)))

rm(X_train_micro,Y_train_micro,C_train_micro,cox_spls_sgpls_fit)
```

`coxspls_sgplsDR` *Fitting a Cox-Model on sparse PLSR components using the (Deviance) Residuals*

Description

This function computes the Cox Model based on PLSR components computed model with

- as the response: the Survival time
- as explanatory variables: Xplan.

It uses the package `sgPLS` to perform group PLSR fit.

Usage

```
coxspls_sgplsDR(Xplan, ...)

## S3 method for class 'formula'
coxspls_sgplsDR(
  Xplan,
  time,
  time2,
  event,
  type,
  origin,
  typeres = "deviance",
  collapse,
  weighted,
  scaleX = TRUE,
  scaleY = TRUE,
  ncomp = min(7, ncol(Xplan)),
  ind.block.x = NULL,
  modepls = "regression",
  keepX,
  alpha.x,
```

```
  upper.lambda = 10^5,
  plot = FALSE,
  allres = FALSE,
  dataXplan = NULL,
  subset,
  weights,
  model_frame = FALSE,
  model_matrix = FALSE,
  contrasts.arg = NULL,
  ...
)

## Default S3 method:
coxspls_sgplsDR(
  Xplan,
  time,
  time2,
  event,
  type,
  origin,
  typeres = "deviance",
  collapse,
  weighted,
  scaleX = TRUE,
  scaleY = TRUE,
  ncomp = min(7, ncol(Xplan)),
  ind.block.x = NULL,
  modepls = "regression",
  keepX,
  alpha.x,
  upper.lambda = 10^5,
  plot = FALSE,
  allres = FALSE,
  ...
)
```

Arguments

Xplan	a formula or a matrix with the eXplanatory variables (training) dataset
...	Arguments to be passed on to survival::coxph.
time	for right censored data, this is the follow up time. For interval data, the first argument is the starting time for the interval.
time2	The status indicator, normally 0=alive, 1=dead. Other choices are TRUE/FALSE (TRUE = death) or 1/2 (2=death). For interval censored data, the status indicator is 0=right censored, 1=event at time, 2=left censored, 3=interval censored. Although unusual, the event indicator can be omitted, in which case all subjects are assumed to have an event.

event	ending time of the interval for interval censored or counting process data only. Intervals are assumed to be open on the left and closed on the right, (start, end]. For counting process data, event indicates whether an event occurred at the end of the interval.
type	character string specifying the type of censoring. Possible values are "right", "left", "counting", "interval", or "interval2". The default is "right" or "counting" depending on whether the time2 argument is absent or present, respectively.
origin	for counting process data, the hazard function origin. This option was intended to be used in conjunction with a model containing time dependent strata in order to align the subjects properly when they cross over from one strata to another, but it has rarely proven useful.
typeres	character string indicating the type of residual desired. Possible values are "martingale", "deviance", "score", "schoenfeld", "dfbeta", "dfbetas", and "scaledsch". Only enough of the string to determine a unique match is required.
collapse	vector indicating which rows to collapse (sum) over. In time-dependent models more than one row data can pertain to a single individual. If there were 4 individuals represented by 3, 1, 2 and 4 rows of data respectively, then collapse=c(1,1,1,2,3,3,4,4,4,4) could be used to obtain per subject rather than per observation residuals.
weighted	if TRUE and the model was fit with case weights, then the weighted residuals are returned.
scaleX	Should the Xplan columns be standardized ?
scaleY	Should the time values be standardized ?
ncomp	The number of components to include in the model. It this is not supplied, min(7,maximal number) components is used.
ind.block.x	a vector of integers describing the grouping of the X-variables. ind.block.x <- c(3,10,15) means that X is structured into 4 groups: X1 to X3; X4 to X10, X11 to X15 and X16 to Xp where p is the number of variables in the X matrix.
modepls	character string. What type of algorithm to use, (partially) matching one of "regression", "canonical". See gPLS for details
keepX	numeric vector of length ncomp, the number of variables to keep in X-loadings. By default all variables are kept in the model.
alpha.x	numeric vector of length ncomp giving the sparsity level applied within each component. Required when ind.block.x is specified.
upper.lambda	numeric value giving the upper bound for the regularized regression penalty used in sgPLS. Defaults to 10^5.
plot	Should the survival function be plotted ?)
allres	FALSE to return only the Cox model and TRUE for additionnal results. See details. Defaults to FALSE.
dataXplan	an optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If not found in dataXplan, the variables are taken from environment(Xplan), typically the environment from which coxpls is called.

subset	an optional vector specifying a subset of observations to be used in the fitting process.
weights	an optional vector of 'prior weights' to be used in the fitting process. Should be NULL or a numeric vector.
model_frame	If TRUE, the model frame is returned.
model_matrix	If TRUE, the model matrix is returned.
contrasts.arg	a list, whose entries are values (numeric matrices, functions or character strings naming functions) to be used as replacement values for the contrasts replacement function and whose names are the names of columns of data containing factors.

Details

If allres=FALSE returns only the final Cox-model. If allres=TRUE returns a list with the PLS components, the final Cox-model and the group PLSR model. allres=TRUE is useful for evluating model prediction accuracy on a test sample.

Value

If allres=FALSE :

cox_spls_sgplsDR
: Final Cox-model.

If allres=TRUE :

tt_spls_sgplsDR
: PLSR components.

cox_spls_sgplsDR
: Final Cox-model.

spls_sgplsDR_mod
: The PLSR model.

Author(s)

Frédéric Bertrand
<frederic.bertrand@lecnam.net>
https://fbertran.github.io/homepage/

References

A group and Sparse Group Partial Least Square approach applied in Genomics context, Liquet Benoit, Lafaye de Micheaux, Boris Hejblum, Rodolphe Thiebaut (2016). Bioinformatics.

Deviance residuals-based sparse PLS and sparse kernel PLS regression for censored data, Philippe Bastien, Frederic Bertrand, Nicolas Meyer and Myriam Maumy-Bertrand (2015), Bioinformatics, 31(3):397-404, doi:10.1093/bioinformatics/btu660.

See Also

coxph, gPLS

Examples

```
data(micro.censure)
data(Xmicro.censure_compl_imp)

X_train_micro <- apply((as.matrix(Xmicro.censure_compl_imp)),
FUN="as.numeric",MARGIN=2)[1:80,]
X_train_micro_df <- data.frame(X_train_micro)
Y_train_micro <- micro.censure$survyear[1:80]
C_train_micro <- micro.censure$DC[1:80]

(cox_spls_sgplsDR_fit=coxspls_sgplsDR(X_train_micro,Y_train_micro,C_train_micro,
ncomp=6,ind.block.x=c(3,10,15), alpha.x = rep(0.95, 6)))
(cox_spls_sgplsDR_fit=coxspls_sgplsDR(~X_train_micro,Y_train_micro,C_train_micro,
ncomp=6,ind.block.x=c(3,10,15), alpha.x = rep(0.95, 6)))
(cox_spls_sgplsDR_fit=coxspls_sgplsDR(~.,Y_train_micro,C_train_micro,ncomp=6,
dataXplan=X_train_micro_df,ind.block.x=c(3,10,15), alpha.x = rep(0.95, 6)))

rm(X_train_micro,Y_train_micro,C_train_micro,cox_spls_sgplsDR_fit)
```

`cv.big_pls_cox` *Cross-validation for big-memory PLS-Cox models*

Description

Performs K-fold cross-validation for models fitted with `big_pls_cox()` or `big_pls_cox_gd()`. The routine mirrors the behaviour of the cross-validation helpers available in the original **plsRcox** package while operating on `big.matrix` inputs.

Usage

```
cv.big_pls_cox(
  data,
  nfold = 5L,
  nt = 5L,
  keepX = NULL,
  givefold,
  allCVcrit = FALSE,
  times.auc = NULL,
  times.prederr = NULL,
  method = c("efron", "breslow"),
  verbose = TRUE,
  ...
)

cv.big_pls_cox_gd(
  data,
  nfold = 5L,
```

```
  nt = NULL,
  keepX = NULL,
  givefold,
  allCVcrit = FALSE,
  times.auc = NULL,
  times.prederr = NULL,
  method = c("efron", "breslow"),
  verbose = TRUE,
  ...
)
```

Arguments

data	A list with entries x, time and status matching the arguments of big_pls_cox() or big_pls_cox_gd(). x can be either a numeric matrix/data frame or a bigmemory::big.matrix.
nfold	Integer giving the number of folds to use.
nt	Number of latent components to evaluate.
keepX	Optional integer vector passed to the modelling function to enforce naive sparsity (see big_pls_cox()).
givefold	Optional list of fold indices. When supplied, it must contain nfold integer vectors whose union is seq_len(nrow(data$x)).
allCVcrit	Logical; when FALSE (default) only the recommended integrated AUC computed with **survivalROC** is returned. When TRUE, the 13 additional criteria from **plsRcox** are also evaluated.
times.auc	Optional time grid used for time-dependent AUC computations. Defaults to an equally spaced grid between zero and the maximum observed time.
times.prederr	Optional time grid used for prediction error curves. Defaults to the same grid as times.auc without the last ten evaluation points to avoid instabilities.
method	Ties handling method passed to survival::coxph.
verbose	Logical; print progress information.
...	Additional arguments forwarded to the underlying modelling function.

Details

The function returns cross-validated estimates for each component (including the null model) using either big_pls_cox() or big_pls_cox_gd(), depending on the engine argument. The implementation reuses the internal indicators (getIndicCV, getIndicCViAUCSurvROCTest) to provide consistent metrics with the legacy **plsRcox** helpers.

Value

A list containing cross-validation summaries. When allCVcrit = FALSE, the list holds

nt	Number of components assessed.
cv.error10	Mean iAUC of **survivalROC** across folds for 0 to nt components.
cv.se10	Estimated standard errors for cv.error10.

folds	Fold assignments.
lambda.min10	Component minimising the cross-validated error.
lambda.1se10	Largest component within one standard error of the optimum.

When allCVcrit = TRUE, the full set of 14 criteria (log partial likelihood, iAUC variants and Brier scores) is returned together with their associated standard errors and one-standard-error selections.

---

cv.coxDKgplsDR *Cross-validating a Direct Kernel group PLS model fitted on the (Deviance) Residuals*

Description

This function cross-validates coxDKgplsDR models.

Usage

```
cv.coxDKgplsDR(
  data,
  method = c("efron", "breslow"),
  nfold = 5,
  nt = 10,
  plot.it = TRUE,
  se = TRUE,
  givefold,
  scaleX = TRUE,
  folddetails = FALSE,
  allCVcrit = FALSE,
  details = FALSE,
  namedataset = "data",
  save = FALSE,
  verbose = TRUE,
  ...
)
```

Arguments

data	A list of three items: • x the explanatory variables passed to coxDKgplsDR's Xplan argument, • time passed to coxDKgplsDR's time argument, • status coxDKgplsDR's status argument.
method	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 here, it is more accurate when dealing with tied death times, and is as efficient computationally.

nfold	The number of folds to use to perform the cross-validation process.
nt	The number of components to include in the model. It this is not supplied, 10 components are fitted.
plot.it	Shall the results be displayed on a plot ?
se	Should standard errors be plotted ?
givefold	Explicit list of omited values in each fold can be provided using this argument.
scaleX	Shall the predictors be standardized ?
folddetails	Should values and completion status for each folds be returned ?
allCVcrit	Should the other 13 CV criteria be evaled and returned ?
details	Should all results of the functions that perform error computations be returned ?
namedataset	Name to use to craft temporary results names
save	Should temporary results be saved ?
verbose	Should some CV details be displayed ?
...	Other arguments to pass to coxDKgplsDR.

Details

It only computes the recommended iAUCSurvROC criterion. Set allCVcrit=TRUE to retrieve the 13 other ones.

Value

nt	The number of components requested
cv.error1	Vector with the mean values, across folds, of, per fold unit, Cross-validated log-partial-likelihood for models with 0 to nt components.
cv.error2	Vector with the mean values, across folds, of, per fold unit, van Houwelingen Cross-validated log-partial-likelihood for models with 0 to nt components.
cv.error3	Vector with the mean values, across folds, of iAUC_CD for models with 0 to nt components.
cv.error4	Vector with the mean values, across folds, of iAUC_hc for models with 0 to nt components.
cv.error5	Vector with the mean values, across folds, of iAUC_sh for models with 0 to nt components.
cv.error6	Vector with the mean values, across folds, of iAUC_Uno for models with 0 to nt components.
cv.error7	Vector with the mean values, across folds, of iAUC_hz.train for models with 0 to nt components.
cv.error8	Vector with the mean values, across folds, of iAUC_hz.test for models with 0 to nt components.
cv.error9	Vector with the mean values, across folds, of iAUC_survivalROC.train for models with 0 to nt components.
cv.error10	Vector with the mean values, across folds, of iAUC_survivalROC.test for models with 0 to nt components.

cv.error11	Vector with the mean values, across folds, of iBrierScore unw for models with 0 to nt components.
cv.error12	Vector with the mean values, across folds, of iSchmidScore (robust BS) unw for models with 0 to nt components.
cv.error13	Vector with the mean values, across folds, of iBrierScore w for models with 0 to nt components.
cv.error14	Vector with the mean values, across folds, of iSchmidScore (robust BS) w for models with 0 to nt components.
cv.se1	Vector with the standard error values, across folds, of, per fold unit, Cross-validated log-partial-likelihood for models with 0 to nt components.
cv.se2	Vector with the standard error values, across folds, of, per fold unit, van Houwelingen Cross-validated log-partial-likelihood for models with 0 to nt components.
cv.se3	Vector with the standard error values, across folds, of iAUC_CD for models with 0 to nt components.
cv.se4	Vector with the standard error values, across folds, of iAUC_hc for models with 0 to nt components.
cv.se5	Vector with the standard error values, across folds, of iAUC_sh for models with 0 to nt components.
cv.se6	Vector with the standard error values, across folds, of iAUC_Uno for models with 0 to nt components.
cv.se7	Vector with the standard error values, across folds, of iAUC_hz.train for models with 0 to nt components.
cv.se8	Vector with the standard error values, across folds, of iAUC_hz.test for models with 0 to nt components.
cv.se9	Vector with the standard error values, across folds, of iAUC_survivalROC.train for models with 0 to nt components.
cv.se10	Vector with the standard error values, across folds, of iAUC_survivalROC.test for models with 0 to nt components.
cv.se11	Vector with the standard error values, across folds, of iBrierScore unw for models with 0 to nt components.
cv.se12	Vector with the standard error values, across folds, of iSchmidScore (robust BS) unw for models with 0 to nt components.
cv.se13	Vector with the standard error values, across folds, of iBrierScore w for models with 0 to nt components.
cv.se14	Vector with the standard error values, across folds, of iSchmidScore (robust BS) w for models with 0 to nt components.
folds	Explicit list of the values that were omited values in each fold.
lambda.min1	Vector with the standard error values, across folds, of, per fold unit, Cross-validated log-partial-likelihood for models with 0 to nt components.
lambda.min2	Vector with the standard error values, across folds, of, per fold unit, van Houwelingen Cross-validated log-partial-likelihood for models with 0 to nt components.
lambda.min1	Optimal Nbr of components, min Cross-validated log-partial-likelihood criterion.

`lambda.se1` Optimal Nbr of components, min+1se Cross-validated log-partial-likelihood criterion.

`lambda.min2` Optimal Nbr of components, min van Houwelingen Cross-validated log-partial-likelihood.

`lambda.se2` Optimal Nbr of components, min+1se van Houwelingen Cross-validated log-partial-likelihood.

`lambda.min3` Optimal Nbr of components, max iAUC_CD criterion.

`lambda.se3` Optimal Nbr of components, max+1se iAUC_CD criterion.

`lambda.min4` Optimal Nbr of components, max iAUC_hc criterion.

`lambda.se4` Optimal Nbr of components, max+1se iAUC_hc criterion.

`lambda.min5` Optimal Nbr of components, max iAUC_sh criterion.

`lambda.se5` Optimal Nbr of components, max+1se iAUC_sh criterion.

`lambda.min6` Optimal Nbr of components, max iAUC_Uno criterion.

`lambda.se6` Optimal Nbr of components, max+1se iAUC_Uno criterion.

`lambda.min7` Optimal Nbr of components, max iAUC_hz.train criterion.

`lambda.se7` Optimal Nbr of components, max+1se iAUC_hz.train criterion.

`lambda.min8` Optimal Nbr of components, max iAUC_hz.test criterion.

`lambda.se8` Optimal Nbr of components, max+1se iAUC_hz.test criterion.

`lambda.min9` Optimal Nbr of components, max iAUC_survivalROC.train criterion.

`lambda.se9` Optimal Nbr of components, max+1se iAUC_survivalROC.train criterion.

`lambda.min10` Optimal Nbr of components, max iAUC_survivalROC.test criterion.

`lambda.se10` Optimal Nbr of components, max+1se iAUC_survivalROC.test criterion.

`lambda.min11` Optimal Nbr of components, min iBrierScore unw criterion.

`lambda.se11` Optimal Nbr of components, min+1se iBrierScore unw criterion.

`lambda.min12` Optimal Nbr of components, min iSchmidScore unw criterion.

`lambda.se12` Optimal Nbr of components, min+1se iSchmidScore unw criterion.

`lambda.min13` Optimal Nbr of components, min iBrierScore w criterion.

`lambda.se13` Optimal Nbr of components, min+1se iBrierScore w criterion.

`lambda.min14` Optimal Nbr of components, min iSchmidScore w criterion.

`lambda.se14` Optimal Nbr of components, min+1se iSchmidScore w criterion.

`errormat1-14` If `details=TRUE`, matrices with the error values for every folds across each of the components and each of the criteria

`completed.cv1-14`
If `details=TRUE`, matrices with logical values for every folds across each of the components and each of the criteria: `TRUE` if the computation was completed and `FALSE` it is failed.

`All_indics` All results of the functions that perform error computation, for each fold, each component and error criterion.

Author(s)

Frédéric Bertrand
<frederic.bertrand@lecnam.net>
https://fbertran.github.io/homepage/

References

plsRcox, Cox-Models in a high dimensional setting in R, Frederic Bertrand, Philippe Bastien, Nicolas Meyer and Myriam Maumy-Bertrand (2014). Proceedings of User2014!, Los Angeles, page 152.

Deviance residuals-based sparse PLS and sparse kernel PLS regression for censored data, Philippe Bastien, Frederic Bertrand, Nicolas Meyer and Myriam Maumy-Bertrand (2015), Bioinformatics, 31(3):397-404, doi:10.1093/bioinformatics/btu660.

Cross validating extensions of kernel, sparse or regular partial least squares regression models to censored data, Bertrand, F., Bastien, Ph. and Maumy-Bertrand, M. (2018), https://arxiv.org/abs/1810.01005.

See Also

See Also coxDKgplsDR

Examples

```
data(micro.censure)
data(Xmicro.censure_compl_imp)
set.seed(123456)
X_train_micro <- apply((as.matrix(Xmicro.censure_compl_imp)),
FUN="as.numeric",MARGIN=2)[1:80,]
X_train_micro_df <- data.frame(X_train_micro)
Y_train_micro <- micro.censure$survyear[1:80]
C_train_micro <- micro.censure$DC[1:80]

#Should be run with a higher value of nt (at least 10)
(cv.coxDKgplsDR.res=cv.coxDKgplsDR(list(x=X_train_micro,time=Y_train_micro,
status=C_train_micro),ind.block.x=c(3,10,15),nt=2))
```

---

cv.coxDKsgplsDR *Cross-validating a Direct Kernel group sparse PLS model fitted on the (Deviance) Residuals*

Description

This function cross-validates coxDKsgplsDR models.

Usage

```
cv.coxDKsgplsDR(
  data,
  method = c("efron", "breslow"),
  nfold = 5,
  nt = 10,
  plot.it = TRUE,
  se = TRUE,
  givefold,
  scaleX = TRUE,
  folddetails = FALSE,
  allCVcrit = FALSE,
  details = FALSE,
  namedataset = "data",
  save = FALSE,
  verbose = TRUE,
  ...
)
```

Arguments

data	A list of three items: • x the explanatory variables passed to coxDKsgplsDR's Xplan argument, • time passed to coxDKsgplsDR's time argument, • status coxDKsgplsDR's status argument.
method	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 here, it is more accurate when dealing with tied death times, and is as efficient computationally.
nfold	The number of folds to use to perform the cross-validation process.
nt	The number of components to include in the model. It this is not supplied, 10 components are fitted.
plot.it	Shall the results be displayed on a plot ?
se	Should standard errors be plotted ?
givefold	Explicit list of omited values in each fold can be provided using this argument.
scaleX	Shall the predictors be standardized ?
folddetails	Should values and completion status for each folds be returned ?
allCVcrit	Should the other 13 CV criteria be evaled and returned ?
details	Should all results of the functions that perform error computations be returned ?
namedataset	Name to use to craft temporary results names
save	Should temporary results be saved ?
verbose	Should some CV details be displayed ?
...	Other arguments to pass to coxDKsgplsDR.

Details

It only computes the recommended iAUCSurvROC criterion. Set `allCVcrit=TRUE` to retrieve the 13 other ones.

Value

`nt`	The number of components requested
`cv.error1`	Vector with the mean values, across folds, of, per fold unit, Cross-validated log-partial-likelihood for models with 0 to nt components.
`cv.error2`	Vector with the mean values, across folds, of, per fold unit, van Houwelingen Cross-validated log-partial-likelihood for models with 0 to nt components.
`cv.error3`	Vector with the mean values, across folds, of iAUC_CD for models with 0 to nt components.
`cv.error4`	Vector with the mean values, across folds, of iAUC_hc for models with 0 to nt components.
`cv.error5`	Vector with the mean values, across folds, of iAUC_sh for models with 0 to nt components.
`cv.error6`	Vector with the mean values, across folds, of iAUC_Uno for models with 0 to nt components.
`cv.error7`	Vector with the mean values, across folds, of iAUC_hz.train for models with 0 to nt components.
`cv.error8`	Vector with the mean values, across folds, of iAUC_hz.test for models with 0 to nt components.
`cv.error9`	Vector with the mean values, across folds, of iAUC_survivalROC.train for models with 0 to nt components.
`cv.error10`	Vector with the mean values, across folds, of iAUC_survivalROC.test for models with 0 to nt components.
`cv.error11`	Vector with the mean values, across folds, of iBrierScore unw for models with 0 to nt components.
`cv.error12`	Vector with the mean values, across folds, of iSchmidScore (robust BS) unw for models with 0 to nt components.
`cv.error13`	Vector with the mean values, across folds, of iBrierScore w for models with 0 to nt components.
`cv.error14`	Vector with the mean values, across folds, of iSchmidScore (robust BS) w for models with 0 to nt components.
`cv.se1`	Vector with the standard error values, across folds, of, per fold unit, Cross-validated log-partial-likelihood for models with 0 to nt components.
`cv.se2`	Vector with the standard error values, across folds, of, per fold unit, van Houwelingen Cross-validated log-partial-likelihood for models with 0 to nt components.
`cv.se3`	Vector with the standard error values, across folds, of iAUC_CD for models with 0 to nt components.
`cv.se4`	Vector with the standard error values, across folds, of iAUC_hc for models with 0 to nt components.

cv.se5	Vector with the standard error values, across folds, of iAUC_sh for models with 0 to nt components.
cv.se6	Vector with the standard error values, across folds, of iAUC_Uno for models with 0 to nt components.
cv.se7	Vector with the standard error values, across folds, of iAUC_hz.train for models with 0 to nt components.
cv.se8	Vector with the standard error values, across folds, of iAUC_hz.test for models with 0 to nt components.
cv.se9	Vector with the standard error values, across folds, of iAUC_survivalROC.train for models with 0 to nt components.
cv.se10	Vector with the standard error values, across folds, of iAUC_survivalROC.test for models with 0 to nt components.
cv.se11	Vector with the standard error values, across folds, of iBrierScore unw for models with 0 to nt components.
cv.se12	Vector with the standard error values, across folds, of iSchmidScore (robust BS) unw for models with 0 to nt components.
cv.se13	Vector with the standard error values, across folds, of iBrierScore w for models with 0 to nt components.
cv.se14	Vector with the standard error values, across folds, of iSchmidScore (robust BS) w for models with 0 to nt components.
folds	Explicit list of the values that were omited values in each fold.
lambda.min1	Vector with the standard error values, across folds, of, per fold unit, Cross-validated log-partial-likelihood for models with 0 to nt components.
lambda.min2	Vector with the standard error values, across folds, of, per fold unit, van Houwelingen Cross-validated log-partial-likelihood for models with 0 to nt components.
lambda.min1	Optimal Nbr of components, min Cross-validated log-partial-likelihood criterion.
lambda.se1	Optimal Nbr of components, min+1se Cross-validated log-partial-likelihood criterion.
lambda.min2	Optimal Nbr of components, min van Houwelingen Cross-validated log-partial-likelihood.
lambda.se2	Optimal Nbr of components, min+1se van Houwelingen Cross-validated log-partial-likelihood.
lambda.min3	Optimal Nbr of components, max iAUC_CD criterion.
lambda.se3	Optimal Nbr of components, max+1se iAUC_CD criterion.
lambda.min4	Optimal Nbr of components, max iAUC_hc criterion.
lambda.se4	Optimal Nbr of components, max+1se iAUC_hc criterion.
lambda.min5	Optimal Nbr of components, max iAUC_sh criterion.
lambda.se5	Optimal Nbr of components, max+1se iAUC_sh criterion.
lambda.min6	Optimal Nbr of components, max iAUC_Uno criterion.
lambda.se6	Optimal Nbr of components, max+1se iAUC_Uno criterion.

lambda.min7	Optimal Nbr of components, max iAUC_hz.train criterion.
lambda.se7	Optimal Nbr of components, max+1se iAUC_hz.train criterion.
lambda.min8	Optimal Nbr of components, max iAUC_hz.test criterion.
lambda.se8	Optimal Nbr of components, max+1se iAUC_hz.test criterion.
lambda.min9	Optimal Nbr of components, max iAUC_survivalROC.train criterion.
lambda.se9	Optimal Nbr of components, max+1se iAUC_survivalROC.train criterion.
lambda.min10	Optimal Nbr of components, max iAUC_survivalROC.test criterion.
lambda.se10	Optimal Nbr of components, max+1se iAUC_survivalROC.test criterion.
lambda.min11	Optimal Nbr of components, min iBrierScore unw criterion.
lambda.se11	Optimal Nbr of components, min+1se iBrierScore unw criterion.
lambda.min12	Optimal Nbr of components, min iSchmidScore unw criterion.
lambda.se12	Optimal Nbr of components, min+1se iSchmidScore unw criterion.
lambda.min13	Optimal Nbr of components, min iBrierScore w criterion.
lambda.se13	Optimal Nbr of components, min+1se iBrierScore w criterion.
lambda.min14	Optimal Nbr of components, min iSchmidScore w criterion.
lambda.se14	Optimal Nbr of components, min+1se iSchmidScore w criterion.
errormat1-14	If details=TRUE, matrices with the error values for every folds across each of the components and each of the criteria
completed.cv1-14	If details=TRUE, matrices with logical values for every folds across each of the components and each of the criteria: TRUE if the computation was completed and FALSE it is failed.
All_indics	All results of the functions that perform error computation, for each fold, each component and error criterion.

Author(s)

Frédéric Bertrand
<frederic.bertrand@lecnam.net>
https://fbertran.github.io/homepage/

References

plsRcox, Cox-Models in a high dimensional setting in R, Frederic Bertrand, Philippe Bastien, Nicolas Meyer and Myriam Maumy-Bertrand (2014). Proceedings of User2014!, Los Angeles, page 152.

Deviance residuals-based sparse PLS and sparse kernel PLS regression for censored data, Philippe Bastien, Frederic Bertrand, Nicolas Meyer and Myriam Maumy-Bertrand (2015), Bioinformatics, 31(3):397-404, doi:10.1093/bioinformatics/btu660.

Cross validating extensions of kernel, sparse or regular partial least squares regression models to censored data, Bertrand, F., Bastien, Ph. and Maumy-Bertrand, M. (2018), https://arxiv.org/abs/1810.01005.

See Also

See Also coxDKsgplsDR

Examples

```
data(micro.censure)
data(Xmicro.censure_compl_imp)
set.seed(123456)
X_train_micro <- apply((as.matrix(Xmicro.censure_compl_imp)),
FUN="as.numeric",MARGIN=2)[1:80,]
X_train_micro_df <- data.frame(X_train_micro)
Y_train_micro <- micro.censure$survyear[1:80]
C_train_micro <- micro.censure$DC[1:80]


#Should be run with a higher value of nt (at least 10)
cv.coxDKsgplsDR.res=cv.coxDKsgplsDR(list(x=X_train_micro,
time=Y_train_micro,status=C_train_micro),ind.block.x=c(3,10,15),
alpha.x = rep(0.95, 6),nt=3,plot.it = FALSE)
cv.coxDKsgplsDR.res
```

---

cv.coxDKspls_sgplsDR *Cross-validating a Direct Kernel sparse PLS model fitted on the (Deviance) Residuals*

Description

This function cross-validates coxDKspls_sgplsDR models.

Usage

```
cv.coxDKspls_sgplsDR(
  data,
  method = c("efron", "breslow"),
  nfold = 5,
  nt = 10,
  plot.it = TRUE,
  se = TRUE,
  givefold,
  scaleX = TRUE,
  folddetails = FALSE,
  allCVcrit = FALSE,
  details = FALSE,
  namedataset = "data",
  save = FALSE,
```

```
  verbose = TRUE,
  ...
)
```

Arguments

data	A list of three items: • x the explanatory variables passed to coxDKspls_sgplsDR's Xplan argument, • time passed to coxDKspls_sgplsDR's time argument, • status coxDKspls_sgplsDR's status argument.
method	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 here, it is more accurate when dealing with tied death times, and is as efficient computationally.
nfold	The number of folds to use to perform the cross-validation process.
nt	The number of components to include in the model. It this is not supplied, 10 components are fitted.
plot.it	Shall the results be displayed on a plot ?
se	Should standard errors be plotted ?
givefold	Explicit list of omited values in each fold can be provided using this argument.
scaleX	Shall the predictors be standardized ?
folddetails	Should values and completion status for each folds be returned ?
allCVcrit	Should the other 13 CV criteria be evaled and returned ?
details	Should all results of the functions that perform error computations be returned ?
namedataset	Name to use to craft temporary results names
save	Should temporary results be saved ?
verbose	Should some CV details be displayed ?
...	Other arguments to pass to coxDKspls_sgplsDR.

Details

It only computes the recommended iAUCSurvROC criterion. Set allCVcrit=TRUE to retrieve the 13 other ones.

Value

nt	The number of components requested
cv.error1	Vector with the mean values, across folds, of, per fold unit, Cross-validated log-partial-likelihood for models with 0 to nt components.
cv.error2	Vector with the mean values, across folds, of, per fold unit, van Houwelingen Cross-validated log-partial-likelihood for models with 0 to nt components.
cv.error3	Vector with the mean values, across folds, of iAUC_CD for models with 0 to nt components.

`cv.error4` Vector with the mean values, across folds, of iAUC_hc for models with 0 to nt components.

`cv.error5` Vector with the mean values, across folds, of iAUC_sh for models with 0 to nt components.

`cv.error6` Vector with the mean values, across folds, of iAUC_Uno for models with 0 to nt components.

`cv.error7` Vector with the mean values, across folds, of iAUC_hz.train for models with 0 to nt components.

`cv.error8` Vector with the mean values, across folds, of iAUC_hz.test for models with 0 to nt components.

`cv.error9` Vector with the mean values, across folds, of iAUC_survivalROC.train for models with 0 to nt components.

`cv.error10` Vector with the mean values, across folds, of iAUC_survivalROC.test for models with 0 to nt components.

`cv.error11` Vector with the mean values, across folds, of iBrierScore unw for models with 0 to nt components.

`cv.error12` Vector with the mean values, across folds, of iSchmidScore (robust BS) unw for models with 0 to nt components.

`cv.error13` Vector with the mean values, across folds, of iBrierScore w for models with 0 to nt components.

`cv.error14` Vector with the mean values, across folds, of iSchmidScore (robust BS) w for models with 0 to nt components.

`cv.se1` Vector with the standard error values, across folds, of, per fold unit, Cross-validated log-partial-likelihood for models with 0 to nt components.

`cv.se2` Vector with the standard error values, across folds, of, per fold unit, van Houwelingen Cross-validated log-partial-likelihood for models with 0 to nt components.

`cv.se3` Vector with the standard error values, across folds, of iAUC_CD for models with 0 to nt components.

`cv.se4` Vector with the standard error values, across folds, of iAUC_hc for models with 0 to nt components.

`cv.se5` Vector with the standard error values, across folds, of iAUC_sh for models with 0 to nt components.

`cv.se6` Vector with the standard error values, across folds, of iAUC_Uno for models with 0 to nt components.

`cv.se7` Vector with the standard error values, across folds, of iAUC_hz.train for models with 0 to nt components.

`cv.se8` Vector with the standard error values, across folds, of iAUC_hz.test for models with 0 to nt components.

`cv.se9` Vector with the standard error values, across folds, of iAUC_survivalROC.train for models with 0 to nt components.

`cv.se10` Vector with the standard error values, across folds, of iAUC_survivalROC.test for models with 0 to nt components.

cv.se11	Vector with the standard error values, across folds, of iBrierScore unw for models with 0 to nt components.
cv.se12	Vector with the standard error values, across folds, of iSchmidScore (robust BS) unw for models with 0 to nt components.
cv.se13	Vector with the standard error values, across folds, of iBrierScore w for models with 0 to nt components.
cv.se14	Vector with the standard error values, across folds, of iSchmidScore (robust BS) w for models with 0 to nt components.
folds	Explicit list of the values that were omited values in each fold.
lambda.min1	Vector with the standard error values, across folds, of, per fold unit, Cross-validated log-partial-likelihood for models with 0 to nt components.
lambda.min2	Vector with the standard error values, across folds, of, per fold unit, van Houwelingen Cross-validated log-partial-likelihood for models with 0 to nt components.
lambda.min1	Optimal Nbr of components, min Cross-validated log-partial-likelihood criterion.
lambda.se1	Optimal Nbr of components, min+1se Cross-validated log-partial-likelihood criterion.
lambda.min2	Optimal Nbr of components, min van Houwelingen Cross-validated log-partial-likelihood.
lambda.se2	Optimal Nbr of components, min+1se van Houwelingen Cross-validated log-partial-likelihood.
lambda.min3	Optimal Nbr of components, max iAUC_CD criterion.
lambda.se3	Optimal Nbr of components, max+1se iAUC_CD criterion.
lambda.min4	Optimal Nbr of components, max iAUC_hc criterion.
lambda.se4	Optimal Nbr of components, max+1se iAUC_hc criterion.
lambda.min5	Optimal Nbr of components, max iAUC_sh criterion.
lambda.se5	Optimal Nbr of components, max+1se iAUC_sh criterion.
lambda.min6	Optimal Nbr of components, max iAUC_Uno criterion.
lambda.se6	Optimal Nbr of components, max+1se iAUC_Uno criterion.
lambda.min7	Optimal Nbr of components, max iAUC_hz.train criterion.
lambda.se7	Optimal Nbr of components, max+1se iAUC_hz.train criterion.
lambda.min8	Optimal Nbr of components, max iAUC_hz.test criterion.
lambda.se8	Optimal Nbr of components, max+1se iAUC_hz.test criterion.
lambda.min9	Optimal Nbr of components, max iAUC_survivalROC.train criterion.
lambda.se9	Optimal Nbr of components, max+1se iAUC_survivalROC.train criterion.
lambda.min10	Optimal Nbr of components, max iAUC_survivalROC.test criterion.
lambda.se10	Optimal Nbr of components, max+1se iAUC_survivalROC.test criterion.
lambda.min11	Optimal Nbr of components, min iBrierScore unw criterion.
lambda.se11	Optimal Nbr of components, min+1se iBrierScore unw criterion.
lambda.min12	Optimal Nbr of components, min iSchmidScore unw criterion.

lambda.se12	Optimal Nbr of components, min+1se iSchmidScore unw criterion.
lambda.min13	Optimal Nbr of components, min iBrierScore w criterion.
lambda.se13	Optimal Nbr of components, min+1se iBrierScore w criterion.
lambda.min14	Optimal Nbr of components, min iSchmidScore w criterion.
lambda.se14	Optimal Nbr of components, min+1se iSchmidScore w criterion.
errormat1-14	If details=TRUE, matrices with the error values for every folds across each of the components and each of the criteria
completed.cv1-14	If details=TRUE, matrices with logical values for every folds across each of the components and each of the criteria: TRUE if the computation was completed and FALSE it is failed.
All_indics	All results of the functions that perform error computation, for each fold, each component and error criterion.

Author(s)

Frédéric Bertrand
<frederic.bertrand@lecnam.net>
https://fbertran.github.io/homepage/

References

plsRcox, Cox-Models in a high dimensional setting in R, Frederic Bertrand, Philippe Bastien, Nicolas Meyer and Myriam Maumy-Bertrand (2014). Proceedings of User2014!, Los Angeles, page 152.

Deviance residuals-based sparse PLS and sparse kernel PLS regression for censored data, Philippe Bastien, Frederic Bertrand, Nicolas Meyer and Myriam Maumy-Bertrand (2015), Bioinformatics, 31(3):397-404, doi:10.1093/bioinformatics/btu660.

Cross validating extensions of kernel, sparse or regular partial least squares regression models to censored data, Bertrand, F., Bastien, Ph. and Maumy-Bertrand, M. (2018), https://arxiv.org/abs/1810.01005.

See Also

See Also coxDKspls_sgplsDR

Examples

```
data(micro.censure)
data(Xmicro.censure_compl_imp)
set.seed(123456)
X_train_micro <- apply((as.matrix(Xmicro.censure_compl_imp)),
FUN="as.numeric",MARGIN=2)[1:80,]
X_train_micro_df <- data.frame(X_train_micro)
Y_train_micro <- micro.censure$survyear[1:80]
C_train_micro <- micro.censure$DC[1:80]
```

```
#Should be run with a higher value of nt (at least 10)
(cv.coxDKspls_sgplsDR.res=cv.coxDKspls_sgplsDR(list(x=X_train_micro,
time=Y_train_micro,status=C_train_micro),ind.block.x=c(3,10,15),
alpha.x = rep(0.95, 3),nt=3))
```

`cv.coxgpls` *Cross-validating a Cox-Model fitted on group PLSR components*

Description

This function cross-validates coxgpls models.

Usage

```
cv.coxgpls(
  data,
  method = c("efron", "breslow"),
  nfold = 5,
  nt = 10,
  plot.it = TRUE,
  se = TRUE,
  givefold,
  scaleX = TRUE,
  folddetails = FALSE,
  allCVcrit = FALSE,
  details = FALSE,
  namedataset = "data",
  save = FALSE,
  verbose = TRUE,
  ...
)
```

Arguments

`data`	A list of three items: • `x` the explanatory variables passed to `coxgpls`'s `Xplan` argument, • `time` passed to `coxgpls`'s `time` argument, • `status` `coxgpls`'s `status` argument.
`method`	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 here, it is more accurate when dealing with tied death times, and is as efficient computationally.
`nfold`	The number of folds to use to perform the cross-validation process.

nt	The number of components to include in the model. It this is not supplied, 10 components are fitted.
plot.it	Shall the results be displayed on a plot ?
se	Should standard errors be plotted ?
givefold	Explicit list of omited values in each fold can be provided using this argument.
scaleX	Shall the predictors be standardized ?
folddetails	Should values and completion status for each folds be returned ?
allCVcrit	Should the other 13 CV criteria be evaled and returned ?
details	Should all results of the functions that perform error computations be returned ?
namedataset	Name to use to craft temporary results names
save	Should temporary results be saved ?
verbose	Should some CV details be displayed ?
...	Other arguments to pass to coxgpls.

Details

It only computes the recommended iAUCSurvROC criterion. Set allCVcrit=TRUE to retrieve the 13 other ones.

Value

nt	The number of components requested
cv.error1	Vector with the mean values, across folds, of, per fold unit, Cross-validated log-partial-likelihood for models with 0 to nt components.
cv.error2	Vector with the mean values, across folds, of, per fold unit, van Houwelingen Cross-validated log-partial-likelihood for models with 0 to nt components.
cv.error3	Vector with the mean values, across folds, of iAUC_CD for models with 0 to nt components.
cv.error4	Vector with the mean values, across folds, of iAUC_hc for models with 0 to nt components.
cv.error5	Vector with the mean values, across folds, of iAUC_sh for models with 0 to nt components.
cv.error6	Vector with the mean values, across folds, of iAUC_Uno for models with 0 to nt components.
cv.error7	Vector with the mean values, across folds, of iAUC_hz.train for models with 0 to nt components.
cv.error8	Vector with the mean values, across folds, of iAUC_hz.test for models with 0 to nt components.
cv.error9	Vector with the mean values, across folds, of iAUC_survivalROC.train for models with 0 to nt components.
cv.error10	Vector with the mean values, across folds, of iAUC_survivalROC.test for models with 0 to nt components.

cv.error11	Vector with the mean values, across folds, of iBrierScore unw for models with 0 to nt components.
cv.error12	Vector with the mean values, across folds, of iSchmidScore (robust BS) unw for models with 0 to nt components.
cv.error13	Vector with the mean values, across folds, of iBrierScore w for models with 0 to nt components.
cv.error14	Vector with the mean values, across folds, of iSchmidScore (robust BS) w for models with 0 to nt components.
cv.se1	Vector with the standard error values, across folds, of, per fold unit, Cross-validated log-partial-likelihood for models with 0 to nt components.
cv.se2	Vector with the standard error values, across folds, of, per fold unit, van Houwelingen Cross-validated log-partial-likelihood for models with 0 to nt components.
cv.se3	Vector with the standard error values, across folds, of iAUC_CD for models with 0 to nt components.
cv.se4	Vector with the standard error values, across folds, of iAUC_hc for models with 0 to nt components.
cv.se5	Vector with the standard error values, across folds, of iAUC_sh for models with 0 to nt components.
cv.se6	Vector with the standard error values, across folds, of iAUC_Uno for models with 0 to nt components.
cv.se7	Vector with the standard error values, across folds, of iAUC_hz.train for models with 0 to nt components.
cv.se8	Vector with the standard error values, across folds, of iAUC_hz.test for models with 0 to nt components.
cv.se9	Vector with the standard error values, across folds, of iAUC_survivalROC.train for models with 0 to nt components.
cv.se10	Vector with the standard error values, across folds, of iAUC_survivalROC.test for models with 0 to nt components.
cv.se11	Vector with the standard error values, across folds, of iBrierScore unw for models with 0 to nt components.
cv.se12	Vector with the standard error values, across folds, of iSchmidScore (robust BS) unw for models with 0 to nt components.
cv.se13	Vector with the standard error values, across folds, of iBrierScore w for models with 0 to nt components.
cv.se14	Vector with the standard error values, across folds, of iSchmidScore (robust BS) w for models with 0 to nt components.
folds	Explicit list of the values that were omited values in each fold.
lambda.min1	Vector with the standard error values, across folds, of, per fold unit, Cross-validated log-partial-likelihood for models with 0 to nt components.
lambda.min2	Vector with the standard error values, across folds, of, per fold unit, van Houwelingen Cross-validated log-partial-likelihood for models with 0 to nt components.
lambda.min1	Optimal Nbr of components, min Cross-validated log-partial-likelihood criterion.

`lambda.se1` Optimal Nbr of components, min+1se Cross-validated log-partial-likelihood criterion.

`lambda.min2` Optimal Nbr of components, min van Houwelingen Cross-validated log-partial-likelihood.

`lambda.se2` Optimal Nbr of components, min+1se van Houwelingen Cross-validated log-partial-likelihood.

`lambda.min3` Optimal Nbr of components, max iAUC_CD criterion.

`lambda.se3` Optimal Nbr of components, max+1se iAUC_CD criterion.

`lambda.min4` Optimal Nbr of components, max iAUC_hc criterion.

`lambda.se4` Optimal Nbr of components, max+1se iAUC_hc criterion.

`lambda.min5` Optimal Nbr of components, max iAUC_sh criterion.

`lambda.se5` Optimal Nbr of components, max+1se iAUC_sh criterion.

`lambda.min6` Optimal Nbr of components, max iAUC_Uno criterion.

`lambda.se6` Optimal Nbr of components, max+1se iAUC_Uno criterion.

`lambda.min7` Optimal Nbr of components, max iAUC_hz.train criterion.

`lambda.se7` Optimal Nbr of components, max+1se iAUC_hz.train criterion.

`lambda.min8` Optimal Nbr of components, max iAUC_hz.test criterion.

`lambda.se8` Optimal Nbr of components, max+1se iAUC_hz.test criterion.

`lambda.min9` Optimal Nbr of components, max iAUC_survivalROC.train criterion.

`lambda.se9` Optimal Nbr of components, max+1se iAUC_survivalROC.train criterion.

`lambda.min10` Optimal Nbr of components, max iAUC_survivalROC.test criterion.

`lambda.se10` Optimal Nbr of components, max+1se iAUC_survivalROC.test criterion.

`lambda.min11` Optimal Nbr of components, min iBrierScore unw criterion.

`lambda.se11` Optimal Nbr of components, min+1se iBrierScore unw criterion.

`lambda.min12` Optimal Nbr of components, min iSchmidScore unw criterion.

`lambda.se12` Optimal Nbr of components, min+1se iSchmidScore unw criterion.

`lambda.min13` Optimal Nbr of components, min iBrierScore w criterion.

`lambda.se13` Optimal Nbr of components, min+1se iBrierScore w criterion.

`lambda.min14` Optimal Nbr of components, min iSchmidScore w criterion.

`lambda.se14` Optimal Nbr of components, min+1se iSchmidScore w criterion.

`errormat1-14` If `details=TRUE`, matrices with the error values for every folds across each of the components and each of the criteria

`completed.cv1-14`
If `details=TRUE`, matrices with logical values for every folds across each of the components and each of the criteria: `TRUE` if the computation was completed and `FALSE` it is failed.

`All_indics` All results of the functions that perform error computation, for each fold, each component and error criterion.

Author(s)

Frédéric Bertrand
<frederic.bertrand@lecnam.net>
https://fbertran.github.io/homepage/

References

plsRcox, Cox-Models in a high dimensional setting in R, Frederic Bertrand, Philippe Bastien, Nicolas Meyer and Myriam Maumy-Bertrand (2014). Proceedings of User2014!, Los Angeles, page 152.

Deviance residuals-based sparse PLS and sparse kernel PLS regression for censored data, Philippe Bastien, Frederic Bertrand, Nicolas Meyer and Myriam Maumy-Bertrand (2015), Bioinformatics, 31(3):397-404, doi:10.1093/bioinformatics/btu660.

Cross validating extensions of kernel, sparse or regular partial least squares regression models to censored data, Bertrand, F., Bastien, Ph. and Maumy-Bertrand, M. (2018), https://arxiv.org/abs/1810.01005.

See Also

See Also coxgpls

Examples

```
data(micro.censure)
data(Xmicro.censure_compl_imp)
set.seed(123456)
X_train_micro <- apply((as.matrix(Xmicro.censure_compl_imp)),
FUN="as.numeric",MARGIN=2)[1:80,]
X_train_micro_df <- data.frame(X_train_micro)
Y_train_micro <- micro.censure$survyear[1:80]
C_train_micro <- micro.censure$DC[1:80]

#Should be run with a higher value of nt (at least 10)
(cv.coxgpls.res=cv.coxgpls(list(x=X_train_micro,time=Y_train_micro,
status=C_train_micro),ind.block.x=c(3,10,15),nt=3))
```

---

cv.coxgplsDR *Cross-validating a Cox-Model fitted on group PLSR components using (Deviance) Residuals*

Description

This function cross-validates coxgplsDR models.

Usage

```
cv.coxgplsDR(
  data,
  method = c("efron", "breslow"),
  nfold = 5,
  nt = 10,
  plot.it = TRUE,
  se = TRUE,
  givefold,
  scaleX = TRUE,
  folddetails = FALSE,
  allCVcrit = FALSE,
  details = FALSE,
  namedataset = "data",
  save = FALSE,
  verbose = TRUE,
  ...
)
```

Arguments

data	A list of three items: • x the explanatory variables passed to coxgpls's Xplan argument, • time passed to coxgpls's time argument, • status coxgpls's status argument.
method	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 here, it is more accurate when dealing with tied death times, and is as efficient computationally.
nfold	The number of folds to use to perform the cross-validation process.
nt	The number of components to include in the model. It this is not supplied, 10 components are fitted.
plot.it	Shall the results be displayed on a plot ?
se	Should standard errors be plotted ?
givefold	Explicit list of omited values in each fold can be provided using this argument.
scaleX	Shall the predictors be standardized ?
folddetails	Should values and completion status for each folds be returned ?
allCVcrit	Should the other 13 CV criteria be evaled and returned ?
details	Should all results of the functions that perform error computations be returned ?
namedataset	Name to use to craft temporary results names
save	Should temporary results be saved ?
verbose	Should some CV details be displayed ?
...	Other arguments to pass to coxgpls.

Details

It only computes the recommended iAUCSurvROC criterion. Set `allCVcrit=TRUE` to retrieve the 13 other ones.

Value

`nt`	The number of components requested
`cv.error1`	Vector with the mean values, across folds, of, per fold unit, Cross-validated log-partial-likelihood for models with 0 to nt components.
`cv.error2`	Vector with the mean values, across folds, of, per fold unit, van Houwelingen Cross-validated log-partial-likelihood for models with 0 to nt components.
`cv.error3`	Vector with the mean values, across folds, of iAUC_CD for models with 0 to nt components.
`cv.error4`	Vector with the mean values, across folds, of iAUC_hc for models with 0 to nt components.
`cv.error5`	Vector with the mean values, across folds, of iAUC_sh for models with 0 to nt components.
`cv.error6`	Vector with the mean values, across folds, of iAUC_Uno for models with 0 to nt components.
`cv.error7`	Vector with the mean values, across folds, of iAUC_hz.train for models with 0 to nt components.
`cv.error8`	Vector with the mean values, across folds, of iAUC_hz.test for models with 0 to nt components.
`cv.error9`	Vector with the mean values, across folds, of iAUC_survivalROC.train for models with 0 to nt components.
`cv.error10`	Vector with the mean values, across folds, of iAUC_survivalROC.test for models with 0 to nt components.
`cv.error11`	Vector with the mean values, across folds, of iBrierScore unw for models with 0 to nt components.
`cv.error12`	Vector with the mean values, across folds, of iSchmidScore (robust BS) unw for models with 0 to nt components.
`cv.error13`	Vector with the mean values, across folds, of iBrierScore w for models with 0 to nt components.
`cv.error14`	Vector with the mean values, across folds, of iSchmidScore (robust BS) w for models with 0 to nt components.
`cv.se1`	Vector with the standard error values, across folds, of, per fold unit, Cross-validated log-partial-likelihood for models with 0 to nt components.
`cv.se2`	Vector with the standard error values, across folds, of, per fold unit, van Houwelingen Cross-validated log-partial-likelihood for models with 0 to nt components.
`cv.se3`	Vector with the standard error values, across folds, of iAUC_CD for models with 0 to nt components.
`cv.se4`	Vector with the standard error values, across folds, of iAUC_hc for models with 0 to nt components.

cv.se5	Vector with the standard error values, across folds, of iAUC_sh for models with 0 to nt components.
cv.se6	Vector with the standard error values, across folds, of iAUC_Uno for models with 0 to nt components.
cv.se7	Vector with the standard error values, across folds, of iAUC_hz.train for models with 0 to nt components.
cv.se8	Vector with the standard error values, across folds, of iAUC_hz.test for models with 0 to nt components.
cv.se9	Vector with the standard error values, across folds, of iAUC_survivalROC.train for models with 0 to nt components.
cv.se10	Vector with the standard error values, across folds, of iAUC_survivalROC.test for models with 0 to nt components.
cv.se11	Vector with the standard error values, across folds, of iBrierScore unw for models with 0 to nt components.
cv.se12	Vector with the standard error values, across folds, of iSchmidScore (robust BS) unw for models with 0 to nt components.
cv.se13	Vector with the standard error values, across folds, of iBrierScore w for models with 0 to nt components.
cv.se14	Vector with the standard error values, across folds, of iSchmidScore (robust BS) w for models with 0 to nt components.
folds	Explicit list of the values that were omited values in each fold.
lambda.min1	Vector with the standard error values, across folds, of, per fold unit, Cross-validated log-partial-likelihood for models with 0 to nt components.
lambda.min2	Vector with the standard error values, across folds, of, per fold unit, van Houwelingen Cross-validated log-partial-likelihood for models with 0 to nt components.
lambda.min1	Optimal Nbr of components, min Cross-validated log-partial-likelihood criterion.
lambda.se1	Optimal Nbr of components, min+1se Cross-validated log-partial-likelihood criterion.
lambda.min2	Optimal Nbr of components, min van Houwelingen Cross-validated log-partial-likelihood.
lambda.se2	Optimal Nbr of components, min+1se van Houwelingen Cross-validated log-partial-likelihood.
lambda.min3	Optimal Nbr of components, max iAUC_CD criterion.
lambda.se3	Optimal Nbr of components, max+1se iAUC_CD criterion.
lambda.min4	Optimal Nbr of components, max iAUC_hc criterion.
lambda.se4	Optimal Nbr of components, max+1se iAUC_hc criterion.
lambda.min5	Optimal Nbr of components, max iAUC_sh criterion.
lambda.se5	Optimal Nbr of components, max+1se iAUC_sh criterion.
lambda.min6	Optimal Nbr of components, max iAUC_Uno criterion.
lambda.se6	Optimal Nbr of components, max+1se iAUC_Uno criterion.

lambda.min7	Optimal Nbr of components, max iAUC_hz.train criterion.
lambda.se7	Optimal Nbr of components, max+1se iAUC_hz.train criterion.
lambda.min8	Optimal Nbr of components, max iAUC_hz.test criterion.
lambda.se8	Optimal Nbr of components, max+1se iAUC_hz.test criterion.
lambda.min9	Optimal Nbr of components, max iAUC_survivalROC.train criterion.
lambda.se9	Optimal Nbr of components, max+1se iAUC_survivalROC.train criterion.
lambda.min10	Optimal Nbr of components, max iAUC_survivalROC.test criterion.
lambda.se10	Optimal Nbr of components, max+1se iAUC_survivalROC.test criterion.
lambda.min11	Optimal Nbr of components, min iBrierScore unw criterion.
lambda.se11	Optimal Nbr of components, min+1se iBrierScore unw criterion.
lambda.min12	Optimal Nbr of components, min iSchmidScore unw criterion.
lambda.se12	Optimal Nbr of components, min+1se iSchmidScore unw criterion.
lambda.min13	Optimal Nbr of components, min iBrierScore w criterion.
lambda.se13	Optimal Nbr of components, min+1se iBrierScore w criterion.
lambda.min14	Optimal Nbr of components, min iSchmidScore w criterion.
lambda.se14	Optimal Nbr of components, min+1se iSchmidScore w criterion.
errormat1-14	If details=TRUE, matrices with the error values for every folds across each of the components and each of the criteria
completed.cv1-14	If details=TRUE, matrices with logical values for every folds across each of the components and each of the criteria: TRUE if the computation was completed and FALSE it is failed.
All_indics	All results of the functions that perform error computation, for each fold, each component and error criterion.

Author(s)

Frédéric Bertrand
<frederic.bertrand@lecnam.net>
https://fbertran.github.io/homepage/

References

plsRcox, Cox-Models in a high dimensional setting in R, Frederic Bertrand, Philippe Bastien, Nicolas Meyer and Myriam Maumy-Bertrand (2014). Proceedings of User2014!, Los Angeles, page 152.

Deviance residuals-based sparse PLS and sparse kernel PLS regression for censored data, Philippe Bastien, Frederic Bertrand, Nicolas Meyer and Myriam Maumy-Bertrand (2015), Bioinformatics, 31(3):397-404, doi:10.1093/bioinformatics/btu660.

Cross validating extensions of kernel, sparse or regular partial least squares regression models to censored data, Bertrand, F., Bastien, Ph. and Maumy-Bertrand, M. (2018), https://arxiv.org/abs/1810.01005.

See Also

See Also coxgpls

Examples

```
data(micro.censure)
data(Xmicro.censure_compl_imp)
set.seed(123456)
X_train_micro <- apply((as.matrix(Xmicro.censure_compl_imp)),
FUN="as.numeric",MARGIN=2)[1:80,]
X_train_micro_df <- data.frame(X_train_micro)
Y_train_micro <- micro.censure$survyear[1:80]
C_train_micro <- micro.censure$DC[1:80]

#Should be run with a higher value of nt (at least 10)
(cv.coxgplsDR.res=cv.coxgplsDR(list(x=X_train_micro,time=Y_train_micro,
status=C_train_micro),ind.block.x=c(3,10,15),nt=3))
```

---

cv.coxsgpls *Cross-validating a Cox-Model fitted on sparse group PLSR components*

Description

This function cross-validates coxsgpls models.

Usage

```
cv.coxsgpls(
  data,
  method = c("efron", "breslow"),
  nfold = 5,
  nt = 10,
  plot.it = TRUE,
  se = TRUE,
  givefold,
  scaleX = TRUE,
  folddetails = FALSE,
  allCVcrit = FALSE,
  details = FALSE,
  namedataset = "data",
  save = FALSE,
  verbose = TRUE,
  ...
)
```

Arguments

data	A list of three items: • x the explanatory variables passed to coxsgpls's Xplan argument, • time passed to coxsgpls's time argument, • status coxsgpls's status argument.
method	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 here, it is more accurate when dealing with tied death times, and is as efficient computationally.
nfold	The number of folds to use to perform the cross-validation process.
nt	The number of components to include in the model. It this is not supplied, 10 components are fitted.
plot.it	Shall the results be displayed on a plot ?
se	Should standard errors be plotted ?
givefold	Explicit list of omited values in each fold can be provided using this argument.
scaleX	Shall the predictors be standardized ?
folddetails	Should values and completion status for each folds be returned ?
allCVcrit	Should the other 13 CV criteria be evaled and returned ?
details	Should all results of the functions that perform error computations be returned ?
namedataset	Name to use to craft temporary results names
save	Should temporary results be saved ?
verbose	Should some CV details be displayed ?
...	Other arguments to pass to coxsgpls.

Details

It only computes the recommended iAUCSurvROC criterion. Set allCVcrit=TRUE to retrieve the 13 other ones.

Value

nt	The number of components requested
cv.error1	Vector with the mean values, across folds, of, per fold unit, Cross-validated log-partial-likelihood for models with 0 to nt components.
cv.error2	Vector with the mean values, across folds, of, per fold unit, van Houwelingen Cross-validated log-partial-likelihood for models with 0 to nt components.
cv.error3	Vector with the mean values, across folds, of iAUC_CD for models with 0 to nt components.
cv.error4	Vector with the mean values, across folds, of iAUC_hc for models with 0 to nt components.
cv.error5	Vector with the mean values, across folds, of iAUC_sh for models with 0 to nt components.

`cv.error6` Vector with the mean values, across folds, of iAUC_Uno for models with 0 to nt components.

`cv.error7` Vector with the mean values, across folds, of iAUC_hz.train for models with 0 to nt components.

`cv.error8` Vector with the mean values, across folds, of iAUC_hz.test for models with 0 to nt components.

`cv.error9` Vector with the mean values, across folds, of iAUC_survivalROC.train for models with 0 to nt components.

`cv.error10` Vector with the mean values, across folds, of iAUC_survivalROC.test for models with 0 to nt components.

`cv.error11` Vector with the mean values, across folds, of iBrierScore unw for models with 0 to nt components.

`cv.error12` Vector with the mean values, across folds, of iSchmidScore (robust BS) unw for models with 0 to nt components.

`cv.error13` Vector with the mean values, across folds, of iBrierScore w for models with 0 to nt components.

`cv.error14` Vector with the mean values, across folds, of iSchmidScore (robust BS) w for models with 0 to nt components.

`cv.se1` Vector with the standard error values, across folds, of, per fold unit, Cross-validated log-partial-likelihood for models with 0 to nt components.

`cv.se2` Vector with the standard error values, across folds, of, per fold unit, van Houwelingen Cross-validated log-partial-likelihood for models with 0 to nt components.

`cv.se3` Vector with the standard error values, across folds, of iAUC_CD for models with 0 to nt components.

`cv.se4` Vector with the standard error values, across folds, of iAUC_hc for models with 0 to nt components.

`cv.se5` Vector with the standard error values, across folds, of iAUC_sh for models with 0 to nt components.

`cv.se6` Vector with the standard error values, across folds, of iAUC_Uno for models with 0 to nt components.

`cv.se7` Vector with the standard error values, across folds, of iAUC_hz.train for models with 0 to nt components.

`cv.se8` Vector with the standard error values, across folds, of iAUC_hz.test for models with 0 to nt components.

`cv.se9` Vector with the standard error values, across folds, of iAUC_survivalROC.train for models with 0 to nt components.

`cv.se10` Vector with the standard error values, across folds, of iAUC_survivalROC.test for models with 0 to nt components.

`cv.se11` Vector with the standard error values, across folds, of iBrierScore unw for models with 0 to nt components.

`cv.se12` Vector with the standard error values, across folds, of iSchmidScore (robust BS) unw for models with 0 to nt components.

cv.se13	Vector with the standard error values, across folds, of iBrierScore w for models with 0 to nt components.
cv.se14	Vector with the standard error values, across folds, of iSchmidScore (robust BS) w for models with 0 to nt components.
folds	Explicit list of the values that were omited values in each fold.
lambda.min1	Vector with the standard error values, across folds, of, per fold unit, Cross-validated log-partial-likelihood for models with 0 to nt components.
lambda.min2	Vector with the standard error values, across folds, of, per fold unit, van Houwelingen Cross-validated log-partial-likelihood for models with 0 to nt components.
lambda.min1	Optimal Nbr of components, min Cross-validated log-partial-likelihood criterion.
lambda.se1	Optimal Nbr of components, min+1se Cross-validated log-partial-likelihood criterion.
lambda.min2	Optimal Nbr of components, min van Houwelingen Cross-validated log-partial-likelihood.
lambda.se2	Optimal Nbr of components, min+1se van Houwelingen Cross-validated log-partial-likelihood.
lambda.min3	Optimal Nbr of components, max iAUC_CD criterion.
lambda.se3	Optimal Nbr of components, max+1se iAUC_CD criterion.
lambda.min4	Optimal Nbr of components, max iAUC_hc criterion.
lambda.se4	Optimal Nbr of components, max+1se iAUC_hc criterion.
lambda.min5	Optimal Nbr of components, max iAUC_sh criterion.
lambda.se5	Optimal Nbr of components, max+1se iAUC_sh criterion.
lambda.min6	Optimal Nbr of components, max iAUC_Uno criterion.
lambda.se6	Optimal Nbr of components, max+1se iAUC_Uno criterion.
lambda.min7	Optimal Nbr of components, max iAUC_hz.train criterion.
lambda.se7	Optimal Nbr of components, max+1se iAUC_hz.train criterion.
lambda.min8	Optimal Nbr of components, max iAUC_hz.test criterion.
lambda.se8	Optimal Nbr of components, max+1se iAUC_hz.test criterion.
lambda.min9	Optimal Nbr of components, max iAUC_survivalROC.train criterion.
lambda.se9	Optimal Nbr of components, max+1se iAUC_survivalROC.train criterion.
lambda.min10	Optimal Nbr of components, max iAUC_survivalROC.test criterion.
lambda.se10	Optimal Nbr of components, max+1se iAUC_survivalROC.test criterion.
lambda.min11	Optimal Nbr of components, min iBrierScore unw criterion.
lambda.se11	Optimal Nbr of components, min+1se iBrierScore unw criterion.
lambda.min12	Optimal Nbr of components, min iSchmidScore unw criterion.
lambda.se12	Optimal Nbr of components, min+1se iSchmidScore unw criterion.
lambda.min13	Optimal Nbr of components, min iBrierScore w criterion.
lambda.se13	Optimal Nbr of components, min+1se iBrierScore w criterion.

lambda.min14	Optimal Nbr of components, min iSchmidScore w criterion.
lambda.se14	Optimal Nbr of components, min+1se iSchmidScore w criterion.
errormat1-14	If details=TRUE, matrices with the error values for every folds across each of the components and each of the criteria
completed.cv1-14	If details=TRUE, matrices with logical values for every folds across each of the components and each of the criteria: TRUE if the computation was completed and FALSE it is failed.
All_indics	All results of the functions that perform error computation, for each fold, each component and error criterion.

Author(s)

Frédéric Bertrand
<frederic.bertrand@lecnam.net>
https://fbertran.github.io/homepage/

References

plsRcox, Cox-Models in a high dimensional setting in R, Frederic Bertrand, Philippe Bastien, Nicolas Meyer and Myriam Maumy-Bertrand (2014). Proceedings of User2014!, Los Angeles, page 152.

Deviance residuals-based sparse PLS and sparse kernel PLS regression for censored data, Philippe Bastien, Frederic Bertrand, Nicolas Meyer and Myriam Maumy-Bertrand (2015), Bioinformatics, 31(3):397-404, doi:10.1093/bioinformatics/btu660.

Cross validating extensions of kernel, sparse or regular partial least squares regression models to censored data, Bertrand, F., Bastien, Ph. and Maumy-Bertrand, M. (2018), https://arxiv.org/abs/1810.01005.

See Also

See Also coxsgpls

Examples

```
data(micro.censure)
data(Xmicro.censure_compl_imp)
set.seed(123456)
X_train_micro <- apply((as.matrix(Xmicro.censure_compl_imp)),
FUN="as.numeric",MARGIN=2)[1:80,]
X_train_micro_df <- data.frame(X_train_micro)
Y_train_micro <- micro.censure$survyear[1:80]
C_train_micro <- micro.censure$DC[1:80]

#Should be run with a higher value of nt (at least 10)
(cv.coxsgpls.res=cv.coxsgpls(list(x=X_train_micro,time=Y_train_micro,
status=C_train_micro),ind.block.x=c(3,10,15), alpha.x = rep(0.95, 6),nt=3))
```

`cv.coxsgplsDR` *Cross-validating a Cox-Model fitted on sparse group PLSR components using (Deviance) Residuals*

Description

This function cross-validates coxsgplsDR models.

Usage

```
cv.coxsgplsDR(
  data,
  method = c("efron", "breslow"),
  nfold = 5,
  nt = 10,
  plot.it = TRUE,
  se = TRUE,
  givefold,
  scaleX = TRUE,
  folddetails = FALSE,
  allCVcrit = FALSE,
  details = FALSE,
  namedataset = "data",
  save = FALSE,
  verbose = TRUE,
  ...
)
```

Arguments

`data`	A list of three items: • `x` the explanatory variables passed to `coxsgplsDR`'s `Xplan` argument, • `time` passed to `coxsgplsDR`'s `time` argument, • `status` `coxsgplsDR`'s `status` argument.
`method`	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 here, it is more accurate when dealing with tied death times, and is as efficient computationally.
`nfold`	The number of folds to use to perform the cross-validation process.
`nt`	The number of components to include in the model. It this is not supplied, 10 components are fitted.
`plot.it`	Shall the results be displayed on a plot ?
`se`	Should standard errors be plotted ?
`givefold`	Explicit list of omited values in each fold can be provided using this argument.

scaleX	Shall the predictors be standardized ?
folddetails	Should values and completion status for each folds be returned ?
allCVcrit	Should the other 13 CV criteria be evaled and returned ?
details	Should all results of the functions that perform error computations be returned ?
namedataset	Name to use to craft temporary results names
save	Should temporary results be saved ?
verbose	Should some CV details be displayed ?
...	Other arguments to pass to coxsgplsDR.

Details

It only computes the recommended iAUCSurvROC criterion. Set allCVcrit=TRUE to retrieve the 13 other ones.

Value

nt	The number of components requested
cv.error1	Vector with the mean values, across folds, of, per fold unit, Cross-validated log-partial-likelihood for models with 0 to nt components.
cv.error2	Vector with the mean values, across folds, of, per fold unit, van Houwelingen Cross-validated log-partial-likelihood for models with 0 to nt components.
cv.error3	Vector with the mean values, across folds, of iAUC_CD for models with 0 to nt components.
cv.error4	Vector with the mean values, across folds, of iAUC_hc for models with 0 to nt components.
cv.error5	Vector with the mean values, across folds, of iAUC_sh for models with 0 to nt components.
cv.error6	Vector with the mean values, across folds, of iAUC_Uno for models with 0 to nt components.
cv.error7	Vector with the mean values, across folds, of iAUC_hz.train for models with 0 to nt components.
cv.error8	Vector with the mean values, across folds, of iAUC_hz.test for models with 0 to nt components.
cv.error9	Vector with the mean values, across folds, of iAUC_survivalROC.train for models with 0 to nt components.
cv.error10	Vector with the mean values, across folds, of iAUC_survivalROC.test for models with 0 to nt components.
cv.error11	Vector with the mean values, across folds, of iBrierScore unw for models with 0 to nt components.
cv.error12	Vector with the mean values, across folds, of iSchmidScore (robust BS) unw for models with 0 to nt components.
cv.error13	Vector with the mean values, across folds, of iBrierScore w for models with 0 to nt components.

cv.error14	Vector with the mean values, across folds, of iSchmidScore (robust BS) w for models with 0 to nt components.
cv.se1	Vector with the standard error values, across folds, of, per fold unit, Cross-validated log-partial-likelihood for models with 0 to nt components.
cv.se2	Vector with the standard error values, across folds, of, per fold unit, van Houwelingen Cross-validated log-partial-likelihood for models with 0 to nt components.
cv.se3	Vector with the standard error values, across folds, of iAUC_CD for models with 0 to nt components.
cv.se4	Vector with the standard error values, across folds, of iAUC_hc for models with 0 to nt components.
cv.se5	Vector with the standard error values, across folds, of iAUC_sh for models with 0 to nt components.
cv.se6	Vector with the standard error values, across folds, of iAUC_Uno for models with 0 to nt components.
cv.se7	Vector with the standard error values, across folds, of iAUC_hz.train for models with 0 to nt components.
cv.se8	Vector with the standard error values, across folds, of iAUC_hz.test for models with 0 to nt components.
cv.se9	Vector with the standard error values, across folds, of iAUC_survivalROC.train for models with 0 to nt components.
cv.se10	Vector with the standard error values, across folds, of iAUC_survivalROC.test for models with 0 to nt components.
cv.se11	Vector with the standard error values, across folds, of iBrierScore unw for models with 0 to nt components.
cv.se12	Vector with the standard error values, across folds, of iSchmidScore (robust BS) unw for models with 0 to nt components.
cv.se13	Vector with the standard error values, across folds, of iBrierScore w for models with 0 to nt components.
cv.se14	Vector with the standard error values, across folds, of iSchmidScore (robust BS) w for models with 0 to nt components.
folds	Explicit list of the values that were omited values in each fold.
lambda.min1	Vector with the standard error values, across folds, of, per fold unit, Cross-validated log-partial-likelihood for models with 0 to nt components.
lambda.min2	Vector with the standard error values, across folds, of, per fold unit, van Houwelingen Cross-validated log-partial-likelihood for models with 0 to nt components.
lambda.min1	Optimal Nbr of components, min Cross-validated log-partial-likelihood criterion.
lambda.se1	Optimal Nbr of components, min+1se Cross-validated log-partial-likelihood criterion.
lambda.min2	Optimal Nbr of components, min van Houwelingen Cross-validated log-partial-likelihood.
lambda.se2	Optimal Nbr of components, min+1se van Houwelingen Cross-validated log-partial-likelihood.

lambda.min3	Optimal Nbr of components, max iAUC_CD criterion.
lambda.se3	Optimal Nbr of components, max+1se iAUC_CD criterion.
lambda.min4	Optimal Nbr of components, max iAUC_hc criterion.
lambda.se4	Optimal Nbr of components, max+1se iAUC_hc criterion.
lambda.min5	Optimal Nbr of components, max iAUC_sh criterion.
lambda.se5	Optimal Nbr of components, max+1se iAUC_sh criterion.
lambda.min6	Optimal Nbr of components, max iAUC_Uno criterion.
lambda.se6	Optimal Nbr of components, max+1se iAUC_Uno criterion.
lambda.min7	Optimal Nbr of components, max iAUC_hz.train criterion.
lambda.se7	Optimal Nbr of components, max+1se iAUC_hz.train criterion.
lambda.min8	Optimal Nbr of components, max iAUC_hz.test criterion.
lambda.se8	Optimal Nbr of components, max+1se iAUC_hz.test criterion.
lambda.min9	Optimal Nbr of components, max iAUC_survivalROC.train criterion.
lambda.se9	Optimal Nbr of components, max+1se iAUC_survivalROC.train criterion.
lambda.min10	Optimal Nbr of components, max iAUC_survivalROC.test criterion.
lambda.se10	Optimal Nbr of components, max+1se iAUC_survivalROC.test criterion.
lambda.min11	Optimal Nbr of components, min iBrierScore unw criterion.
lambda.se11	Optimal Nbr of components, min+1se iBrierScore unw criterion.
lambda.min12	Optimal Nbr of components, min iSchmidScore unw criterion.
lambda.se12	Optimal Nbr of components, min+1se iSchmidScore unw criterion.
lambda.min13	Optimal Nbr of components, min iBrierScore w criterion.
lambda.se13	Optimal Nbr of components, min+1se iBrierScore w criterion.
lambda.min14	Optimal Nbr of components, min iSchmidScore w criterion.
lambda.se14	Optimal Nbr of components, min+1se iSchmidScore w criterion.
errormat1-14	If details=TRUE, matrices with the error values for every folds across each of the components and each of the criteria
completed.cv1-14	If details=TRUE, matrices with logical values for every folds across each of the components and each of the criteria: TRUE if the computation was completed and FALSE it is failed.
All_indics	All results of the functions that perform error computation, for each fold, each component and error criterion.

Author(s)

Frédéric Bertrand
<frederic.bertrand@lecnam.net>
https://fbertran.github.io/homepage/

References

plsRcox, Cox-Models in a high dimensional setting in R, Frederic Bertrand, Philippe Bastien, Nicolas Meyer and Myriam Maumy-Bertrand (2014). Proceedings of User2014!, Los Angeles, page 152.

Deviance residuals-based sparse PLS and sparse kernel PLS regression for censored data, Philippe Bastien, Frederic Bertrand, Nicolas Meyer and Myriam Maumy-Bertrand (2015), Bioinformatics, 31(3):397-404, doi:10.1093/bioinformatics/btu660.

Cross validating extensions of kernel, sparse or regular partial least squares regression models to censored data, Bertrand, F., Bastien, Ph. and Maumy-Bertrand, M. (2018), https://arxiv.org/abs/1810.01005.

See Also

See Also coxsgplsDR

Examples

```
data(micro.censure)
data(Xmicro.censure_compl_imp)
set.seed(123456)
X_train_micro <- apply((as.matrix(Xmicro.censure_compl_imp)),
FUN="as.numeric",MARGIN=2)[1:80,]
X_train_micro_df <- data.frame(X_train_micro)
Y_train_micro <- micro.censure$survyear[1:80]
C_train_micro <- micro.censure$DC[1:80]

#Should be run with a higher value of nt (at least 10)
(cv.coxsgplsDR.res=cv.coxsgplsDR(list(x=X_train_micro,time=Y_train_micro,
status=C_train_micro),ind.block.x=c(3,10,15), alpha.x = rep(0.95, 6),nt=2))
```

---

cv.coxspls_sgpls *Cross-validating a Cox-Model fitted on sparse PLSR components*

Description

This function cross-validates coxspls_sgpls models.

Usage

```
cv.coxspls_sgpls(
  data,
  method = c("efron", "breslow"),
  nfold = 5,
  nt = 10,
  plot.it = TRUE,
```

```
  se = TRUE,
  givefold,
  scaleX = TRUE,
  folddetails = FALSE,
  allCVcrit = FALSE,
  details = FALSE,
  namedataset = "data",
  save = FALSE,
  verbose = TRUE,
  ...
)
```

Arguments

data	A list of three items: • x the explanatory variables passed to coxspls_sgpls's Xplan argument, • time passed to coxspls_sgpls's time argument, • status coxspls_sgpls's status argument.
method	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 here, it is more accurate when dealing with tied death times, and is as efficient computationally.
nfold	The number of folds to use to perform the cross-validation process.
nt	The number of components to include in the model. It this is not supplied, 10 components are fitted.
plot.it	Shall the results be displayed on a plot ?
se	Should standard errors be plotted ?
givefold	Explicit list of omited values in each fold can be provided using this argument.
scaleX	Shall the predictors be standardized ?
folddetails	Should values and completion status for each folds be returned ?
allCVcrit	Should the other 13 CV criteria be evaled and returned ?
details	Should all results of the functions that perform error computations be returned ?
namedataset	Name to use to craft temporary results names
save	Should temporary results be saved ?
verbose	Should some CV details be displayed ?
...	Other arguments to pass to coxspls_sgpls.

Details

It only computes the recommended iAUCSurvROC criterion. Set allCVcrit=TRUE to retrieve the 13 other ones.

Value

nt	The number of components requested
cv.error1	Vector with the mean values, across folds, of, per fold unit, Cross-validated log-partial-likelihood for models with 0 to nt components.
cv.error2	Vector with the mean values, across folds, of, per fold unit, van Houwelingen Cross-validated log-partial-likelihood for models with 0 to nt components.
cv.error3	Vector with the mean values, across folds, of iAUC_CD for models with 0 to nt components.
cv.error4	Vector with the mean values, across folds, of iAUC_hc for models with 0 to nt components.
cv.error5	Vector with the mean values, across folds, of iAUC_sh for models with 0 to nt components.
cv.error6	Vector with the mean values, across folds, of iAUC_Uno for models with 0 to nt components.
cv.error7	Vector with the mean values, across folds, of iAUC_hz.train for models with 0 to nt components.
cv.error8	Vector with the mean values, across folds, of iAUC_hz.test for models with 0 to nt components.
cv.error9	Vector with the mean values, across folds, of iAUC_survivalROC.train for models with 0 to nt components.
cv.error10	Vector with the mean values, across folds, of iAUC_survivalROC.test for models with 0 to nt components.
cv.error11	Vector with the mean values, across folds, of iBrierScore unw for models with 0 to nt components.
cv.error12	Vector with the mean values, across folds, of iSchmidScore (robust BS) unw for models with 0 to nt components.
cv.error13	Vector with the mean values, across folds, of iBrierScore w for models with 0 to nt components.
cv.error14	Vector with the mean values, across folds, of iSchmidScore (robust BS) w for models with 0 to nt components.
cv.se1	Vector with the standard error values, across folds, of, per fold unit, Cross-validated log-partial-likelihood for models with 0 to nt components.
cv.se2	Vector with the standard error values, across folds, of, per fold unit, van Houwelingen Cross-validated log-partial-likelihood for models with 0 to nt components.
cv.se3	Vector with the standard error values, across folds, of iAUC_CD for models with 0 to nt components.
cv.se4	Vector with the standard error values, across folds, of iAUC_hc for models with 0 to nt components.
cv.se5	Vector with the standard error values, across folds, of iAUC_sh for models with 0 to nt components.
cv.se6	Vector with the standard error values, across folds, of iAUC_Uno for models with 0 to nt components.

cv.se7	Vector with the standard error values, across folds, of iAUC_hz.train for models with 0 to nt components.
cv.se8	Vector with the standard error values, across folds, of iAUC_hz.test for models with 0 to nt components.
cv.se9	Vector with the standard error values, across folds, of iAUC_survivalROC.train for models with 0 to nt components.
cv.se10	Vector with the standard error values, across folds, of iAUC_survivalROC.test for models with 0 to nt components.
cv.se11	Vector with the standard error values, across folds, of iBrierScore unw for models with 0 to nt components.
cv.se12	Vector with the standard error values, across folds, of iSchmidScore (robust BS) unw for models with 0 to nt components.
cv.se13	Vector with the standard error values, across folds, of iBrierScore w for models with 0 to nt components.
cv.se14	Vector with the standard error values, across folds, of iSchmidScore (robust BS) w for models with 0 to nt components.
folds	Explicit list of the values that were omited values in each fold.
lambda.min1	Vector with the standard error values, across folds, of, per fold unit, Cross-validated log-partial-likelihood for models with 0 to nt components.
lambda.min2	Vector with the standard error values, across folds, of, per fold unit, van Houwelingen Cross-validated log-partial-likelihood for models with 0 to nt components.
lambda.min1	Optimal Nbr of components, min Cross-validated log-partial-likelihood criterion.
lambda.se1	Optimal Nbr of components, min+1se Cross-validated log-partial-likelihood criterion.
lambda.min2	Optimal Nbr of components, min van Houwelingen Cross-validated log-partial-likelihood.
lambda.se2	Optimal Nbr of components, min+1se van Houwelingen Cross-validated log-partial-likelihood.
lambda.min3	Optimal Nbr of components, max iAUC_CD criterion.
lambda.se3	Optimal Nbr of components, max+1se iAUC_CD criterion.
lambda.min4	Optimal Nbr of components, max iAUC_hc criterion.
lambda.se4	Optimal Nbr of components, max+1se iAUC_hc criterion.
lambda.min5	Optimal Nbr of components, max iAUC_sh criterion.
lambda.se5	Optimal Nbr of components, max+1se iAUC_sh criterion.
lambda.min6	Optimal Nbr of components, max iAUC_Uno criterion.
lambda.se6	Optimal Nbr of components, max+1se iAUC_Uno criterion.
lambda.min7	Optimal Nbr of components, max iAUC_hz.train criterion.
lambda.se7	Optimal Nbr of components, max+1se iAUC_hz.train criterion.
lambda.min8	Optimal Nbr of components, max iAUC_hz.test criterion.

lambda.se8	Optimal Nbr of components, max+1se iAUC_hz.test criterion.
lambda.min9	Optimal Nbr of components, max iAUC_survivalROC.train criterion.
lambda.se9	Optimal Nbr of components, max+1se iAUC_survivalROC.train criterion.
lambda.min10	Optimal Nbr of components, max iAUC_survivalROC.test criterion.
lambda.se10	Optimal Nbr of components, max+1se iAUC_survivalROC.test criterion.
lambda.min11	Optimal Nbr of components, min iBrierScore unw criterion.
lambda.se11	Optimal Nbr of components, min+1se iBrierScore unw criterion.
lambda.min12	Optimal Nbr of components, min iSchmidScore unw criterion.
lambda.se12	Optimal Nbr of components, min+1se iSchmidScore unw criterion.
lambda.min13	Optimal Nbr of components, min iBrierScore w criterion.
lambda.se13	Optimal Nbr of components, min+1se iBrierScore w criterion.
lambda.min14	Optimal Nbr of components, min iSchmidScore w criterion.
lambda.se14	Optimal Nbr of components, min+1se iSchmidScore w criterion.
errormat1-14	If details=TRUE, matrices with the error values for every folds across each of the components and each of the criteria
completed.cv1-14	If details=TRUE, matrices with logical values for every folds across each of the components and each of the criteria: TRUE if the computation was completed and FALSE it is failed.
All_indics	All results of the functions that perform error computation, for each fold, each component and error criterion.

Author(s)

Frédéric Bertrand
<frederic.bertrand@lecnam.net>
https://fbertran.github.io/homepage/

References

plsRcox, Cox-Models in a high dimensional setting in R, Frederic Bertrand, Philippe Bastien, Nicolas Meyer and Myriam Maumy-Bertrand (2014). Proceedings of User2014!, Los Angeles, page 152.

Deviance residuals-based sparse PLS and sparse kernel PLS regression for censored data, Philippe Bastien, Frederic Bertrand, Nicolas Meyer and Myriam Maumy-Bertrand (2015), Bioinformatics, 31(3):397-404, doi:10.1093/bioinformatics/btu660.

Cross validating extensions of kernel, sparse or regular partial least squares regression models to censored data, Bertrand, F., Bastien, Ph. and Maumy-Bertrand, M. (2018), https://arxiv.org/abs/1810.01005.

See Also

See Also coxspls_sgpls

Examples

```
data(micro.censure)
data(Xmicro.censure_compl_imp)
set.seed(123456)
X_train_micro <- apply((as.matrix(Xmicro.censure_compl_imp)),
FUN="as.numeric",MARGIN=2)[1:80,]
X_train_micro_df <- data.frame(X_train_micro)
Y_train_micro <- micro.censure$survyear[1:80]
C_train_micro <- micro.censure$DC[1:80]

#Should be run with a higher value of nt (at least 10)
(cv.coxspls_sgpls.res=cv.coxspls_sgpls(list(x=X_train_micro,
time=Y_train_micro,status=C_train_micro),ind.block.x=c(3,10,15),
alpha.x = rep(0.95, 6),nt=3))
```

`cv.coxspls_sgplsDR` *Cross-validating a Cox-Model fitted on sparse PLSR components components using (Deviance) Residuals*

Description

This function cross-validates coxspls_sgplsDR models.

Usage

```
cv.coxspls_sgplsDR(
  data,
  method = c("efron", "breslow"),
  nfold = 5,
  nt = 10,
  plot.it = TRUE,
  se = TRUE,
  givefold,
  scaleX = TRUE,
  folddetails = FALSE,
  allCVcrit = FALSE,
  details = FALSE,
  namedataset = "data",
  save = FALSE,
  verbose = TRUE,
  ...
)
```

Arguments

data	A list of three items: • x the explanatory variables passed to coxspls_sgplsDR's Xplan argument, • time passed to coxspls_sgplsDR's time argument, • status coxspls_sgplsDR's status argument.
method	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 here, it is more accurate when dealing with tied death times, and is as efficient computationally.
nfold	The number of folds to use to perform the cross-validation process.
nt	The number of components to include in the model. It this is not supplied, 10 components are fitted.
plot.it	Shall the results be displayed on a plot ?
se	Should standard errors be plotted ?
givefold	Explicit list of omited values in each fold can be provided using this argument.
scaleX	Shall the predictors be standardized ?
folddetails	Should values and completion status for each folds be returned ?
allCVcrit	Should the other 13 CV criteria be evaled and returned ?
details	Should all results of the functions that perform error computations be returned ?
namedataset	Name to use to craft temporary results names
save	Should temporary results be saved ?
verbose	Should some CV details be displayed ?
...	Other arguments to pass to coxspls_sgplsDR.

Details

It only computes the recommended iAUCSurvROC criterion. Set allCVcrit=TRUE to retrieve the 13 other ones.

Value

nt	The number of components requested
cv.error1	Vector with the mean values, across folds, of, per fold unit, Cross-validated log-partial-likelihood for models with 0 to nt components.
cv.error2	Vector with the mean values, across folds, of, per fold unit, van Houwelingen Cross-validated log-partial-likelihood for models with 0 to nt components.
cv.error3	Vector with the mean values, across folds, of iAUC_CD for models with 0 to nt components.
cv.error4	Vector with the mean values, across folds, of iAUC_hc for models with 0 to nt components.
cv.error5	Vector with the mean values, across folds, of iAUC_sh for models with 0 to nt components.

`cv.error6` Vector with the mean values, across folds, of iAUC_Uno for models with 0 to nt components.

`cv.error7` Vector with the mean values, across folds, of iAUC_hz.train for models with 0 to nt components.

`cv.error8` Vector with the mean values, across folds, of iAUC_hz.test for models with 0 to nt components.

`cv.error9` Vector with the mean values, across folds, of iAUC_survivalROC.train for models with 0 to nt components.

`cv.error10` Vector with the mean values, across folds, of iAUC_survivalROC.test for models with 0 to nt components.

`cv.error11` Vector with the mean values, across folds, of iBrierScore unw for models with 0 to nt components.

`cv.error12` Vector with the mean values, across folds, of iSchmidScore (robust BS) unw for models with 0 to nt components.

`cv.error13` Vector with the mean values, across folds, of iBrierScore w for models with 0 to nt components.

`cv.error14` Vector with the mean values, across folds, of iSchmidScore (robust BS) w for models with 0 to nt components.

`cv.se1` Vector with the standard error values, across folds, of, per fold unit, Cross-validated log-partial-likelihood for models with 0 to nt components.

`cv.se2` Vector with the standard error values, across folds, of, per fold unit, van Houwelingen Cross-validated log-partial-likelihood for models with 0 to nt components.

`cv.se3` Vector with the standard error values, across folds, of iAUC_CD for models with 0 to nt components.

`cv.se4` Vector with the standard error values, across folds, of iAUC_hc for models with 0 to nt components.

`cv.se5` Vector with the standard error values, across folds, of iAUC_sh for models with 0 to nt components.

`cv.se6` Vector with the standard error values, across folds, of iAUC_Uno for models with 0 to nt components.

`cv.se7` Vector with the standard error values, across folds, of iAUC_hz.train for models with 0 to nt components.

`cv.se8` Vector with the standard error values, across folds, of iAUC_hz.test for models with 0 to nt components.

`cv.se9` Vector with the standard error values, across folds, of iAUC_survivalROC.train for models with 0 to nt components.

`cv.se10` Vector with the standard error values, across folds, of iAUC_survivalROC.test for models with 0 to nt components.

`cv.se11` Vector with the standard error values, across folds, of iBrierScore unw for models with 0 to nt components.

`cv.se12` Vector with the standard error values, across folds, of iSchmidScore (robust BS) unw for models with 0 to nt components.

cv.se13	Vector with the standard error values, across folds, of iBrierScore w for models with 0 to nt components.
cv.se14	Vector with the standard error values, across folds, of iSchmidScore (robust BS) w for models with 0 to nt components.
folds	Explicit list of the values that were omited values in each fold.
lambda.min1	Vector with the standard error values, across folds, of, per fold unit, Cross-validated log-partial-likelihood for models with 0 to nt components.
lambda.min2	Vector with the standard error values, across folds, of, per fold unit, van Houwelingen Cross-validated log-partial-likelihood for models with 0 to nt components.
lambda.min1	Optimal Nbr of components, min Cross-validated log-partial-likelihood criterion.
lambda.se1	Optimal Nbr of components, min+1se Cross-validated log-partial-likelihood criterion.
lambda.min2	Optimal Nbr of components, min van Houwelingen Cross-validated log-partial-likelihood.
lambda.se2	Optimal Nbr of components, min+1se van Houwelingen Cross-validated log-partial-likelihood.
lambda.min3	Optimal Nbr of components, max iAUC_CD criterion.
lambda.se3	Optimal Nbr of components, max+1se iAUC_CD criterion.
lambda.min4	Optimal Nbr of components, max iAUC_hc criterion.
lambda.se4	Optimal Nbr of components, max+1se iAUC_hc criterion.
lambda.min5	Optimal Nbr of components, max iAUC_sh criterion.
lambda.se5	Optimal Nbr of components, max+1se iAUC_sh criterion.
lambda.min6	Optimal Nbr of components, max iAUC_Uno criterion.
lambda.se6	Optimal Nbr of components, max+1se iAUC_Uno criterion.
lambda.min7	Optimal Nbr of components, max iAUC_hz.train criterion.
lambda.se7	Optimal Nbr of components, max+1se iAUC_hz.train criterion.
lambda.min8	Optimal Nbr of components, max iAUC_hz.test criterion.
lambda.se8	Optimal Nbr of components, max+1se iAUC_hz.test criterion.
lambda.min9	Optimal Nbr of components, max iAUC_survivalROC.train criterion.
lambda.se9	Optimal Nbr of components, max+1se iAUC_survivalROC.train criterion.
lambda.min10	Optimal Nbr of components, max iAUC_survivalROC.test criterion.
lambda.se10	Optimal Nbr of components, max+1se iAUC_survivalROC.test criterion.
lambda.min11	Optimal Nbr of components, min iBrierScore unw criterion.
lambda.se11	Optimal Nbr of components, min+1se iBrierScore unw criterion.
lambda.min12	Optimal Nbr of components, min iSchmidScore unw criterion.
lambda.se12	Optimal Nbr of components, min+1se iSchmidScore unw criterion.
lambda.min13	Optimal Nbr of components, min iBrierScore w criterion.
lambda.se13	Optimal Nbr of components, min+1se iBrierScore w criterion.

lambda.min14	Optimal Nbr of components, min iSchmidScore w criterion.
lambda.se14	Optimal Nbr of components, min+1se iSchmidScore w criterion.
errormat1-14	If details=TRUE, matrices with the error values for every folds across each of the components and each of the criteria
completed.cv1-14	If details=TRUE, matrices with logical values for every folds across each of the components and each of the criteria: TRUE if the computation was completed and FALSE it is failed.
All_indics	All results of the functions that perform error computation, for each fold, each component and error criterion.

Author(s)

Frédéric Bertrand
<frederic.bertrand@lecnam.net>
https://fbertran.github.io/homepage/

References

plsRcox, Cox-Models in a high dimensional setting in R, Frederic Bertrand, Philippe Bastien, Nicolas Meyer and Myriam Maumy-Bertrand (2014). Proceedings of User2014!, Los Angeles, page 152.

Deviance residuals-based sparse PLS and sparse kernel PLS regression for censored data, Philippe Bastien, Frederic Bertrand, Nicolas Meyer and Myriam Maumy-Bertrand (2015), Bioinformatics, 31(3):397-404, doi:10.1093/bioinformatics/btu660.

Cross validating extensions of kernel, sparse or regular partial least squares regression models to censored data, Bertrand, F., Bastien, Ph. and Maumy-Bertrand, M. (2018), https://arxiv.org/abs/1810.01005.

See Also

See Also coxspls_sgplsDR

Examples

```
data(micro.censure)
data(Xmicro.censure_compl_imp)
set.seed(123456)
X_train_micro <- apply((as.matrix(Xmicro.censure_compl_imp)),
FUN="as.numeric",MARGIN=2)[1:80,]
X_train_micro_df <- data.frame(X_train_micro)
Y_train_micro <- micro.censure$survyear[1:80]
C_train_micro <- micro.censure$DC[1:80]

#Should be run with a higher value of nt (at least 10)
(cv.coxspls_sgplsDR.res=cv.coxspls_sgplsDR(list(x=X_train_micro,
time=Y_train_micro,status=C_train_micro),ind.block.x=c(3,10,15),
alpha.x = rep(0.95, 6),nt=3))
```

dataCox *Cox Proportional Hazards Model Data Generation From Weibull Distribution*

Description

Function dataCox generaters random survivaldata from Weibull distribution (with parameters lambda and rho for given input x data, model coefficients beta and censoring rate for censoring that comes from exponential distribution with parameter cens.rate.

Usage

```
dataCox(n, lambda, rho, x, beta, cens.rate)
```

Arguments

n	Number of observations to generate.
lambda	lambda parameter for Weibull distribution.
rho	rho parameter for Weibull distribution.
x	A data.frame with an input data to generate the survival times for.
beta	True model coefficients.
cens.rate	Parameter for exponential distribution, which is responsible for censoring.

Details

For each observation true survival time is generated and a censroing time. If censoring time is less then survival time, then the survival time is returned and a status of observations is set to 0 which means the observation had censored time. If the survival time is less than censoring time, then for this observation the true survival time is returned and the status of this observation is set to 1 which means that the event has been noticed.

Value

A data.frame containing columns:

- id an integer.
- time survival times.
- status observation status (event occured (1) or not (0)).
- x a data.frame with an input data to generate the survival times for.

References

http://onlinelibrary.wiley.com/doi/10.1002/sim.2059/abstract

Generating survival times to simulate Cox proportional hazards models, 2005 by Ralf Bender, Thomas Augustin, Maria Blettner.

Examples

```
x <- matrix(sample(0:1, size = 20000, replace = TRUE), ncol = 2)
dCox <- dataCox(10^4, lambda = 3, rho = 2, x,
beta = c(1,3), cens.rate = 5)
```

---

dCox_sim *Simulated survival dataset for Cox models*

Description

The `dCox_sim` dataset contains simulated survival times, censoring indicators and two binary covariates for demonstrating the Cox-related procedures included in **bigPLScox**.

Format

A data frame with 10000 observations on the following 5 variables.

id observation identifier

time simulated survival time

status event indicator (1 = event, 0 = censored)

x.1 first binary covariate

x.2 second binary covariate

Examples

```
data(dCox_sim)
with(dCox_sim, table(status))
```

---

gd_diagnostics *Extract Diagnostics from a big_pls_cox_gd Model*

Description

Extract Diagnostics from a big_pls_cox_gd Model

Usage

```
gd_diagnostics(object)
```

Arguments

`object`	A model returned by `big_pls_cox_gd()`.

Value

A list with log-likelihood, step sizes, gradient norms.

Examples

```
library(bigmemory)
set.seed(1)
n <- 50
p <- 10
X <- bigmemory::as.big.matrix(matrix(rnorm(n * p), n, p))
time <- rexp(n, rate = 0.1)
status <- rbinom(n, 1, 0.7)
fit <- big_pls_cox_gd(X, time, status, ncomp = 3, max_iter = 200)
str(fit)
head(fit$scores)
gd_diagnostics(fit)
```

---

micro.censure *Microsat features and survival times*

Description

This dataset provides Microsat specifications and survival times.

Format

A data frame with 117 observations on the following 43 variables.

numpat a factor with levels B1006 B1017 B1028 B1031 B1046 B1059 B1068 B1071 B1102 B1115 B1124 B1139 B1157 B1161 B1164 B1188 B1190 B1192 B1203 B1211 B1221 B1225 B1226 B1227 B1237 B1251 B1258 B1266 B1271 B1282 B1284 B1285 B1286 B1287 B1290 B1292 B1298 B1302 B1304 B1310 B1319 B1327 B1353 B1357 B1363 B1368 B1372 B1373 B1379 B1388 B1392 B1397 B1403 B1418 B1421t1 B1421t2 B1448 B1451 B1455 B1460 B1462 B1466 B1469 B1493 B1500 B1502 B1519 B1523 B1529 B1530 B1544 B1548 B500 B532 B550 B558 B563 B582 B605 B609 B634 B652 B667 B679 B701 B722 B728 B731 B736 B739 B744 B766 B771 B777 B788 B800 B836 B838 B841 B848 B871 B873 B883 B889 B912 B924 B925 B927 B938 B952 B954 B955 B968 B972 B976 B982 B984

D18S61 a numeric vector

D17S794 a numeric vector

D13S173 a numeric vector

D20S107 a numeric vector

TP53 a numeric vector

D9S171 a numeric vector

D8S264 a numeric vector

D5S346 a numeric vector

D22S928 a numeric vector

D18S53 a numeric vector

D1S225 a numeric vector

D3S1282 a numeric vector

D15S127 a numeric vector

D1S305 a numeric vector

D1S207 a numeric vector

D2S138 a numeric vector

D16S422 a numeric vector

D9S179 a numeric vector

D10S191 a numeric vector

D4S394 a numeric vector

D1S197 a numeric vector

D6S264 a numeric vector

D14S65 a numeric vector

D17S790 a numeric vector

D5S430 a numeric vector

D3S1283 a numeric vector

D4S414 a numeric vector

D8S283 a numeric vector

D11S916 a numeric vector

D2S159 a numeric vector

D16S408 a numeric vector

D6S275 a numeric vector

D10S192 a numeric vector

sexe a numeric vector

Agediag a numeric vector

Siege a numeric vector

T a numeric vector

N a numeric vector

M a numeric vector

STADE a factor with levels `0` `1` `2` `3` `4`

survyear a numeric vector

DC a numeric vector

Source

Allelotyping identification of genomic alterations in rectal chromosomally unstable tumors without preoperative treatment, #' Benoît Romain, Agnès Neuville, Nicolas Meyer, Cécile Brigand, Serge Rohr, Anne Schneider, Marie-Pierre Gaub and Dominique Guenot, *BMC Cancer 2010*, 10:561, doi:10.1186/1471-2407-10-561.

References

plsRcox, Cox-Models in a high dimensional setting in R, Frederic Bertrand, Philippe Bastien, Nicolas Meyer and Myriam Maumy-Bertrand (2014). Proceedings of User2014!, Los Angeles, page 152.

Deviance residuals-based sparse PLS and sparse kernel PLS regression for censored data, Philippe Bastien, Frederic Bertrand, Nicolas Meyer and Myriam Maumy-Bertrand (2015), Bioinformatics, 31(3):397-404, doi:10.1093/bioinformatics/btu660.

Examples

```
data(micro.censure)
Y_train_micro <- micro.censure$survyear[1:80]
C_train_micro <- micro.censure$DC[1:80]
Y_test_micro <- micro.censure$survyear[81:117]
C_test_micro <- micro.censure$DC[81:117]
rm(Y_train_micro,C_train_micro,Y_test_micro,C_test_micro)
```

---

partialbigSurvSGDv0 *Incremental Survival Model Fitting with Pre-Scaled Data*

Description

Loads a previously scaled design matrix and continues the stochastic gradient optimisation for a subset of variables.

Usage

```
partialbigSurvSGDv0(
  name.col,
  datapath,
  ncores = 1,
  resBigscale,
  bigmemory.flag = FALSE,
  parallel.flag = FALSE,
  inf.mth = "none"
)
```

Arguments

`name.col`	Character vector containing the column names that should be included in the partial fit.
`datapath`	File system path or connection where the big-memory backing file for the scaled design matrix is stored.

ncores	Number of processor cores allocated to the partial fitting procedure. Defaults to 1.
resBigscale	Result object returned by bigscale containing scaling statistics to be reused. By default the helper reuses the globally cached resultsBigscale object created by bigscale.
bigmemory.flag	Logical flag determining whether big-memory backed matrices are used when loading and updating the design matrix. Defaults to FALSE.
parallel.flag	Logical flag toggling the use of parallelised stochastic gradient updates. Defaults to FALSE.
inf.mth	Inference method requested for the partial fit, such as "none", "asymptotic", or bootstrap summaries. Defaults to "none".

Value

Either a numeric vector of log hazard-ratio coefficients or, when inference is requested, a matrix whose columns correspond to the inferred coefficient summaries for each penalisation setting.

See Also

bigscale(), bigSurvSGD.na.omit() and bigSurvSGD.

Examples

```
data(micro.censure, package = "bigPLScox")
surv_data <- stats::na.omit(
  micro.censure[, c("survyear", "DC", "sexe", "Agediag")]
)
scaled <- bigscale(
  survival::Surv(survyear, DC) ~ .,
  data = surv_data,
  norm.method = "standardize",
  batch.size = 16
)
datapath <- tempfile(fileext = ".csv")
utils::write.csv(surv_data, datapath, row.names = FALSE)

continued <- partialbigSurvSGDv0(
  name.col = c("Agediag", "sexe"),
  datapath = datapath,
  ncores = 1,
  resBigscale = scaled,
  bigmemory.flag = FALSE,
  parallel.flag = FALSE,
  inf.mth = "none"
)
```

plot.big_pls_cox *Plot method for big_pls_cox objects*

Description

Plot method for big_pls_cox objects

Usage

```
## S3 method for class 'big_pls_cox'
plot(
  x,
  which = c("scores", "loadings", "risk_groups"),
  comps = c(1, 2),
  groups = NULL,
  breaks = 3,
  ...
)
```

Arguments

x	A big_pls_cox object.
which	Type of plot: "scores", "loadings", or "risk_groups".
comps	Components to use (for "scores" and "loadings").
groups	Optional grouping factor for scores plot; if NULL and status available, groups are derived from event status.
breaks	Number of risk groups for "risk_groups" (default 3).
...	Further graphical arguments (passed to base plotting functions).

plot.big_pls_cox_gd *Plot method for big_pls_cox_gd objects*

Description

Plot method for big_pls_cox_gd objects

Basic diagnostic plots for big_pls_cox_gd

Usage

```
## S3 method for class 'big_pls_cox_gd'
plot(x, what = c("loglik", "gradient", "steps", "coef", "risk", "all"), ...)

## S3 method for class 'big_pls_cox_gd'
plot(x, what = c("loglik", "gradient", "steps", "coef", "risk", "all"), ...)
```

Arguments

x	A big_pls_cox object.
what	Character string specifying what to plot. Typically "scores", "loadings" or "coefficients".
...	Further graphical arguments (passed to base plotting functions).

---

predict.big_pls_cox *Predict method for big-memory PLS-Cox models*

Description

Predict method for big-memory PLS-Cox models

Usage

```
## S3 method for class 'big_pls_cox'
predict(
  object,
  newdata = NULL,
  type = c("link", "risk", "response", "components"),
  comps = NULL,
  coef = NULL,
  ...
)
```

Arguments

object	A model fitted with big_pls_cox().
newdata	Optional matrix, data frame or bigmemory::big.matrix containing predictors to project on the latent space. When NULL the training scores are used.
type	Type of prediction: "link" for the linear predictor, "risk" or "response" for the exponential of the linear predictor, or "components" to obtain latent scores.
comps	Integer vector indicating which components to use. Defaults to all available components.
coef	Optional coefficient vector overriding the fitted Cox model coefficients.
...	Unused.

Value

Depending on type, either a numeric vector of predictions or a matrix of component scores.

References

Maumy, M., Bertrand, F. (2023). PLS models and their extension for big data. Joint Statistical Meetings (JSM 2023), Toronto, ON, Canada.

Maumy, M., Bertrand, F. (2023). bigPLS: Fitting and cross-validating PLS-based Cox models to censored big data. BioC2023 — The Bioconductor Annual Conference, Dana-Farber Cancer Institute, Boston, MA, USA. Poster. https://doi.org/10.7490/f1000research.1119546.1

Bastien, P., Bertrand, F., Meyer, N., & Maumy-Bertrand, M. (2015). Deviance residuals-based sparse PLS and sparse kernel PLS for censored data. *Bioinformatics*, 31(3), 397–404. doi:10.1093/bioinformatics/btu660

Bertrand, F., Bastien, P., Meyer, N., & Maumy-Bertrand, M. (2014). PLS models for censored data. In *Proceedings of UseR! 2014* (p. 152).

See Also

`big_pls_cox()`, `big_pls_cox_gd()`, `select_ncomp()`, `computeDR()`.

`predict.big_pls_cox_fast`

Predictions for fast big PLS–Cox fits

Description

Predictions for fast big PLS–Cox fits

Usage

```
## S3 method for class 'big_pls_cox_fast'
predict(
  object,
  newdata = NULL,
  type = c("link", "risk", "response", "components"),
  comps = NULL,
  coef = NULL,
  ...
)
```

Arguments

`object`	An object of class `"big_pls_cox_fast"`.
`newdata`	Optional matrix or `big.matrix` with predictors.
`type`	Type of prediction: `"link"`, `"risk"`, `"response"`, or `"components"`.
`comps`	Optional components to use. Defaults to all.
`coef`	Optional coefficient vector. Defaults to `object$coefficients`.
`...`	Ignored.

predict.big_pls_cox_gd *Predict method for big_pls_cox_gd*

Description

Predict method for big_pls_cox_gd

Usage

```
## S3 method for class 'big_pls_cox_gd'
predict(
  object,
  newdata = NULL,
  type = c("link", "risk", "response", "components"),
  comps = NULL,
  coef = NULL,
  ...
)
```

Arguments

object	A big_pls_cox_gd model.
newdata	A numeric matrix or big.matrix. If NULL, uses training scores.
type	One of "link", "risk", "response", "components".
comps	Integer vector of components to use. Default: all.
coef	Optional coefficient vector. If NULL, uses coef(object$cox_fit).
...	Unused.

predict_cox_pls *Predict survival summaries from legacy Cox-PLS fits*

Description

These methods extend `stats::predict()` for Cox models fitted with the original PLS engines exposed by `coxgpls()`, `coxsgpls()`, and their deviance-residual or kernel variants. They provide access to latent component scores alongside linear predictors and risk estimates, ensuring consistent behaviour with the newer big-memory solvers.

Usage

```
## S3 method for class 'coxgpls'
predict(
  object,
  newdata = NULL,
  type = c("link", "risk", "response", "components"),
  comps = NULL,
  coef = NULL,
  ...
)

## S3 method for class 'coxgplsDR'
predict(
  object,
  newdata = NULL,
  type = c("link", "risk", "response", "components"),
  comps = NULL,
  coef = NULL,
  ...
)

## S3 method for class 'coxsgpls'
predict(
  object,
  newdata = NULL,
  type = c("link", "risk", "response", "components"),
  comps = NULL,
  coef = NULL,
  ...
)

## S3 method for class 'coxsgplsDR'
predict(
  object,
  newdata = NULL,
  type = c("link", "risk", "response", "components"),
  comps = NULL,
  coef = NULL,
  ...
)

## S3 method for class 'coxspls_sgpls'
predict(
  object,
  newdata = NULL,
  type = c("link", "risk", "response", "components"),
  comps = NULL,
  coef = NULL,
```

```
  ...
)

## S3 method for class 'coxDKgplsDR'
predict(
  object,
  newdata = NULL,
  type = c("link", "risk", "response", "components"),
  comps = NULL,
  coef = NULL,
  ...
)

## S3 method for class 'coxDKsgplsDR'
predict(
  object,
  newdata = NULL,
  type = c("link", "risk", "response", "components"),
  comps = NULL,
  coef = NULL,
  ...
)

## S3 method for class 'coxDKspls_sgplsDR'
predict(
  object,
  newdata = NULL,
  type = c("link", "risk", "response", "components"),
  comps = NULL,
  coef = NULL,
  ...
)
```

Arguments

`object`	A fitted model returned by `coxgpls()`, `coxsgpls()`, `coxspls_sgpls()`, or any of their deviance-residual/kernel counterparts with `allres = TRUE`.
`newdata`	Optional matrix or data frame of predictors. When `NULL`, the training components stored in `object` are reused.
`type`	Type of prediction requested: `"link"` for linear predictors, `"risk"`/`"response"` for exponentiated scores, or `"components"` to return latent PLS scores.
`comps`	Optional integer vector specifying which latent components to retain. Defaults to all available components.
`coef`	Optional coefficient vector overriding the Cox model coefficients stored in `object`.
`...`	Unused arguments for future extensions.

Value

When `type` is `"components"`, a matrix of latent scores; otherwise a numeric vector containing the requested prediction with names inherited from the supplied data.

References

Bastien, P., Bertrand, F., Meyer, N., & Maumy-Bertrand, M. (2015). Deviance residuals-based sparse PLS and sparse kernel PLS for censored data. *Bioinformatics*, 31(3), 397–404. doi:10.1093/bioinformatics/btu660

Bertrand, F., Bastien, P., & Maumy-Bertrand, M. (2018). Cross validating extensions of kernel, sparse or regular partial least squares regression models to censored data. https://arxiv.org/abs/1810.01005.

See Also

`coxgpls()`, `coxsgpls()`, `coxspls_sgpls()`, `coxDKgplsDR()`, `predict.big_pls_cox()`, `computeDR()`.

Examples

```
if (requireNamespace("survival", quietly = TRUE)) {
  data(micro.censure, package = "bigPLScox")
  data(Xmicro.censure_compl_imp, package = "bigPLScox")

  X <- as.matrix(Xmicro.censure_compl_imp[1:60, 1:10])
  time <- micro.censure$survyear[1:60]
  status <- micro.censure$DC[1:60]

  set.seed(321)
  fit <- coxgpls(
    Xplan = X,
    time = time,
    status = status,
    ncomp = 2,
    allres = TRUE
  )

  predict(fit, newdata = X[1:5, ], type = "risk")
  head(predict(fit, type = "components"))
}
```

---

predict_pls_latent *Predict responses and latent scores from PLS fits*

Description

These prediction helpers reconstruct the response matrix and latent component scores for partial least squares (PLS) models fitted inside the Cox-PLS toolbox. They support group PLS, sparse PLS, sparse-group PLS, and classical PLS models created by `sgPLS::gPLS()`, `sgPLS::sPLS()`, `sgPLS::sgPLS()`, or `plsRcox::pls.cox()`.

Usage

```
## S3 method for class 'gPLS'
predict(object, newdata, scale.X = TRUE, scale.Y = TRUE, ...)

## S3 method for class 'pls.cox'
predict(object, newdata, scale.X = TRUE, scale.Y = TRUE, ...)

## S3 method for class 'sPLS'
predict(object, newdata, scale.X = TRUE, scale.Y = TRUE, ...)

## S3 method for class 'sgPLS'
predict(object, newdata, scale.X = TRUE, scale.Y = TRUE, ...)
```

Arguments

`object`	A fitted PLS model returned by `sgPLS::gPLS()`, `sgPLS::sPLS()`, `sgPLS::sgPLS()`, or `plsRcox::pls.cox()`.
`newdata`	Numeric matrix or data frame with the same number of columns as the training design matrix used when fitting `object`.
`scale.X`, `scale.Y`	Logical flags indicating whether the predictors and responses supplied in `newdata` should be centred and scaled according to the training statistics stored in `object`.
`...`	Unused arguments included for compatibility with the generic `stats::predict()` signature.

Value

A list containing reconstructed responses, latent component scores, and regression coefficients. The exact elements depend on the specific PLS algorithm but always include components named `predict`, `variates`, and `B.hat`.

References

Bastien, P., Bertrand, F., Meyer, N., & Maumy-Bertrand, M. (2015). Deviance residuals-based sparse PLS and sparse kernel PLS for censored data. *Bioinformatics*, 31(3), 397–404. doi:10.1093/bioinformatics/btu660

See Also

`coxgpls()`, `coxsgpls()`, `coxspls_sgpls()`, and `coxDKgplsDR()` for Cox model wrappers that return PLS fits using these prediction methods.

Examples

```
n <- 100
sigma.gamma <- 1
sigma.e <- 1.5
p <- 400
q <- 500
```

```
theta.x1 <- c(rep(1, 15), rep(0, 5), rep(-1, 15), rep(0, 5), rep(1.5,15),
              rep(0, 5), rep(-1.5, 15), rep(0, 325))
theta.x2 <- c(rep(0, 320), rep(1, 15), rep(0, 5), rep(-1, 15), rep(0, 5),
              rep(1.5, 15), rep(0, 5), rep(-1.5, 15), rep(0, 5))
theta.y1 <- 1
theta.y2 <- 1

Sigmax <- matrix(0, nrow = p, ncol = p)
diag(Sigmax) <- sigma.e ^ 2
Sigmay <- matrix(0,nrow = 1, ncol = 1)
diag(Sigmay) <- sigma.e ^ 2

set.seed(125)

gam1 <- rnorm(n)
gam2 <- rnorm(n)

X <- matrix(c(gam1, gam2), ncol = 2, byrow = FALSE) %*% matrix(c(theta.x1, theta.x2),
 nrow = 2, byrow = TRUE) + mvtnorm::rmvnorm(n, mean = rep(0, p), sigma =
 Sigmax, method = "svd")
Y <- matrix(c(gam1, gam2), ncol = 2, byrow = FALSE) %*% matrix(c(theta.y1, theta.y2),
nrow = 2, byrow = TRUE) + rnorm(n,0,sd=sigma.e)

ind.block.x <- seq(20, 380, 20)

model.gPLS <- sgPLS::gPLS(X, Y, ncomp = 2, mode = "regression", keepX = c(4, 4),
                    keepY = c(4, 4), ind.block.x = ind.block.x)
head(predict(model.gPLS, newdata = X)$variates)
```

`print.big_pls_cox` *Print method for big_pls_cox objects*

Description

Print method for big_pls_cox objects

Usage

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

Arguments

`x`	An object of class `big_pls_cox`.
`...`	Further arguments passed to or from other methods.

print.big_pls_cox_gd *Print method for big_pls_cox_gd objects*

Description

Print method for big_pls_cox_gd objects

Usage

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

Arguments

x	An object of class big_pls_cox_gd.
...	Further arguments passed to or from other methods.

print.summary.big_pls_cox_fast *Print method for summary.big_pls_cox_fast objects*

Description

Print method for summary.big_pls_cox_fast objects

Usage

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

Arguments

x	An object of class summary.big_pls_cox_fast.
...	Further arguments passed to or from other methods.

`sim_data` *Simulated dataset*

Description

This dataset provides explantory variables simulations and censoring status.

Format

A data frame with 1000 observations on the following 11 variables.

status a binary vector

X1 a numeric vector

X2 a numeric vector

X3 a numeric vector

X4 a numeric vector

X5 a numeric vector

X6 a numeric vector

X7 a numeric vector

X8 a numeric vector

X9 a numeric vector

X10 a numeric vector

References

Maumy, M., Bertrand, F. (2023). PLS models and their extension for big data. Joint Statistical Meetings (JSM 2023), Toronto, ON, Canada.

Maumy, M., Bertrand, F. (2023). bigPLS: Fitting and cross-validating PLS-based Cox models to censored big data. BioC2023 — The Bioconductor Annual Conference, Dana-Farber Cancer Institute, Boston, MA, USA. Poster. https://doi.org/10.7490/f1000research.1119546.1

Bastien, P., Bertrand, F., Meyer, N., and Maumy-Bertrand, M. (2015). Deviance residuals-based sparse PLS and sparse kernel PLS for binary classification and survival analysis. *BMC Bioinformatics*, 16, 211.

Examples

```
data(sim_data)
X_sim_data_train <- sim_data[1:800,2:11]
C_sim_data_train <- sim_data$status[1:800]
X_sim_data_test <- sim_data[801:1000,2:11]
C_sim_data_test <- sim_data$status[801:1000]
rm(X_sim_data_train,C_sim_data_train,X_sim_data_test,C_sim_data_test)
```

summary.big_pls_cox *Summary for big_pls_cox objects*

Description

Summary for big_pls_cox objects

Usage

```
## S3 method for class 'big_pls_cox'
summary(object, digits = 3, ...)
```

Arguments

object	A big_pls_cox object.
digits	Number of digits to print.
...	Unused, for S3 compatibility.

summary.big_pls_cox_fast *Summary for big_pls_cox objects*

Description

Summary for big_pls_cox objects

Usage

```
## S3 method for class 'big_pls_cox_fast'
summary(object, ...)
```

Arguments

object	A big_pls_cox object.
...	Not used.

summary.big_pls_cox_gd *Summary for big_pls_cox_gd objects*

Description

Summary for big_pls_cox_gd objects

Usage

```
## S3 method for class 'big_pls_cox_gd'
summary(object, digits = 3, ...)
```

Arguments

object	A big_pls_cox_gd object.
digits	Number of digits to print.
...	Unused, for S3 compatibility.

Xmicro.censure_compl_imp *Imputed Microsat features*

Description

This dataset provides imputed microsat specifications. Imputations were computed using Multivariate Imputation by Chained Equations (MICE) using predictive mean matching for the numeric columns, logistic regression imputation for the binary data or the factors with 2 levels and polytomous regression imputation for categorical data i.e. factors with three or more levels.

Format

A data frame with 117 observations on the following 40 variables.

D18S61 a numeric vector

D17S794 a numeric vector

D13S173 a numeric vector

D20S107 a numeric vector

TP53 a numeric vector

D9S171 a numeric vector

D8S264 a numeric vector

D5S346 a numeric vector

D22S928 a numeric vector

D18S53 a numeric vector

D1S225 a numeric vector

D3S1282 a numeric vector

D15S127 a numeric vector

D1S305 a numeric vector

D1S207 a numeric vector

D2S138 a numeric vector

D16S422 a numeric vector

D9S179 a numeric vector

D10S191 a numeric vector

D4S394 a numeric vector

D1S197 a numeric vector

D6S264 a numeric vector

D14S65 a numeric vector

D17S790 a numeric vector

D5S430 a numeric vector

D3S1283 a numeric vector

D4S414 a numeric vector

D8S283 a numeric vector

D11S916 a numeric vector

D2S159 a numeric vector

D16S408 a numeric vector

D6S275 a numeric vector

D10S192 a numeric vector

sexe a numeric vector

Agediag a numeric vector

Siege a numeric vector

T a numeric vector

N a numeric vector

M a numeric vector

STADE a factor with levels `0` `1` `2` `3` `4`

Source

Allelotyping identification of genomic alterations in rectal chromosomally unstable tumors without preoperative treatment, Benoît Romain, Agnès Neuville, Nicolas Meyer, Cécile Brigand, Serge Rohr, Anne Schneider, Marie-Pierre Gaub and Dominique Guenot, *BMC Cancer 2010*, 10:561, doi:10.1186/1471-2407-10-561.

References

plsRcox, Cox-Models in a high dimensional setting in R, Frederic Bertrand, Philippe Bastien, Nicolas Meyer and Myriam Maumy-Bertrand (2014). Proceedings of User2014!, Los Angeles, page 152.

Deviance residuals-based sparse PLS and sparse kernel PLS regression for censored data, Philippe Bastien, Frederic Bertrand, Nicolas Meyer and Myriam Maumy-Bertrand (2015), Bioinformatics, 31(3):397-404, doi:10.1093/bioinformatics/btu660.

Examples

```
data(Xmicro.censure_compl_imp)
X_train_micro <- Xmicro.censure_compl_imp[1:80,]
X_test_micro <- Xmicro.censure_compl_imp[81:117,]
rm(X_train_micro,X_test_micro)
```

Index

∗ **datasets**
dCox_sim, 106
micro.censure, 107
sim_data, 121
Xmicro.censure_compl_imp, 123
∗ **methods**
bigmatrix-operations, 11
∗ **models**
computeDR, 18
coxDKgplsDR, 22
coxDKsgplsDR, 27
coxDKspls_sgplsDR, 32
coxgpls, 37
coxgplsDR, 41
coxsgpls, 45
coxsgplsDR, 49
coxspls_sgpls, 53
coxspls_sgplsDR, 57
cv.coxDKgplsDR, 63
cv.coxDKsgplsDR, 67
cv.coxDKspls_sgplsDR, 72
cv.coxgpls, 77
cv.coxgplsDR, 81
cv.coxsgpls, 86
cv.coxsgplsDR, 91
cv.coxspls_sgpls, 95
cv.coxspls_sgplsDR, 100
∗ **regression**
computeDR, 18
coxDKgplsDR, 22
coxDKsgplsDR, 27
coxDKspls_sgplsDR, 32
coxgpls, 37
coxgplsDR, 41
coxsgpls, 45
coxsgplsDR, 49
coxspls_sgpls, 53
coxspls_sgplsDR, 57
cv.coxDKgplsDR, 63
cv.coxDKsgplsDR, 67
cv.coxDKspls_sgplsDR, 72
cv.coxgpls, 77
cv.coxgplsDR, 81
cv.coxsgpls, 86
cv.coxsgplsDR, 91
cv.coxspls_sgpls, 95
cv.coxspls_sgplsDR, 100
%*%,big.matrix,big.matrix-method *(bigmatrix-operations)*, 11
%*%,big.matrix,matrix-method *(bigmatrix-operations)*, 11
%*%,matrix,big.matrix-method *(bigmatrix-operations)*, 11
%*%, *11*

align_big_plscox, 4
Arith,big.matrix,big.matrix-method *(bigmatrix-operations)*, 11
Arith,big.matrix,matrix-method *(bigmatrix-operations)*, 11
Arith,big.matrix,numeric-method *(bigmatrix-operations)*, 11
Arith,matrix,big.matrix-method *(bigmatrix-operations)*, 11
Arith,numeric,big.matrix-method *(bigmatrix-operations)*, 11
Arithmetic, *11*
as.data.frame, *25*, *30*, *34*, *39*, *43*, *47*, *51*, *55*, *59*

bench::mark, *21*, *22*
benchmark_deviance_residuals, *21*
benchmark_deviance_residuals *(cox_deviance_residuals)*, 21
benchmark_deviance_residuals(), *22*
big.matrix, *11*
big_pls_cox, 5
big_pls_cox(), *4*, *10*, *17*, *61*, *62*, *112*, *113*
big_pls_cox_fast, 6

big_pls_cox_fast(), *7*, *9*
big_pls_cox_gd, 7
big_pls_cox_gd(), *4*, *6*, *17*, *61*, *62*, *113*
big_pls_cox_transform, 10
bigalgebra::dadd(), *12*
bigalgebra::daxpy(), *12*
bigalgebra::dgemm(), *12*
bigmatrix-operations, 11
bigmemory::big.matrix, *5*, *7*, *8*, *18*, *19*, *21*, *62*, *112*
bigmemory::big.matrix(), *12*
bigPLScox *(*bigPLScox-package*)*, 3
bigPLScox-package, 3
bigscale, 12, *17*, *110*
bigscale(), *110*
bigSurvSGD, *17*, *110*
bigSurvSGD.na.omit, 14
bigSurvSGD.na.omit(), *13*, *110*

component_information, 17
component_information(), *18*
component_information, *(*component_information*)*, 17
component_information.big_pls_cox *(*component_information*)*, 17
component_information.big_pls_cox, *(*component_information*)*, 17
component_information.big_pls_cox_gd *(*component_information*)*, 17
component_information.big_pls_cox_gd, *(*component_information*)*, 17
computeDR, 18
computeDR(), *6*, *10*, *113*, *117*
cox_deviance_details *(*cox_deviance_residuals*)*, 21
cox_deviance_details(), *22*
cox_deviance_residuals, 21
cox_deviance_residuals(), *21*, *22*
cox_deviance_residuals_big *(*cox_deviance_residuals*)*, 21
cox_deviance_residuals_big(), *22*
cox_partial_deviance_big *(*cox_deviance_residuals*)*, 21
cox_partial_deviance_big(), *22*
coxDKgplsDR, 22, *63*, *64*, *67*
coxDKgplsDR(), *117*, *118*
coxDKsgplsDR, 27, *67*, *68*, *72*
coxDKspls_sgplsDR, 32, *72*, *73*, *76*
coxgpls, 37, *77*, *78*, *81*, *82*, *86*
coxgpls(), *114*, *116–118*
coxgplsDR, 41, *81*
coxph, *20*, *26*, *31*, *36*, *40*, *44*, *48*, *52*, *56*, *60*
coxsgpls, 45, *86*, *87*, *90*
coxsgpls(), *114*, *116–118*
coxsgplsDR, 49, *91*, *92*, *95*
coxspls_sgpls, 53, *95*, *96*, *99*
coxspls_sgpls(), *116–118*
coxspls_sgplsDR, 57, *100*, *101*, *104*
cv.big_pls_cox, 61
cv.big_pls_cox_gd *(*cv.big_pls_cox*)*, 61
cv.coxDKgplsDR, 63
cv.coxDKsgplsDR, 67
cv.coxDKspls_sgplsDR, 72
cv.coxgpls, 77
cv.coxgplsDR, 81
cv.coxsgpls, 86
cv.coxsgplsDR, 91
cv.coxspls_sgpls, 95
cv.coxspls_sgplsDR, 100

dataCox, 105
dCox_sim, 106

gd_diagnostics, 106
gPLS, *25*, *26*, *29*, *31*, *34*, *36*, *39*, *40*, *43*, *44*, *47*, *48*, *51*, *52*, *55*, *56*, *59*, *60*

kernels, *25*, *30*, *35*

micro.censure, 107

partialbigSurvSGDv0, *17*, 109
plot.big_pls_cox, 111
plot.big_pls_cox_gd, 111
plsRcox::pls.cox(), *117*, *118*
predict.big_pls_cox, 112
predict.big_pls_cox(), *6*, *10*, *117*
predict.big_pls_cox_fast, 113
predict.big_pls_cox_gd, 114
predict.coxDKgplsDR *(*predict_cox_pls*)*, 114
predict.coxDKsgplsDR *(*predict_cox_pls*)*, 114
predict.coxDKspls_sgplsDR *(*predict_cox_pls*)*, 114
predict.coxgpls *(*predict_cox_pls*)*, 114
predict.coxgplsDR *(*predict_cox_pls*)*, 114
predict.coxsgpls *(*predict_cox_pls*)*, 114

predict.coxsgplsDR *(*predict_cox_pls*)*, 114
predict.coxspls_sgpls *(*predict_cox_pls*)*, 114
predict.gPLS *(*predict_pls_latent*)*, 117
predict.pls.cox *(*predict_pls_latent*)*, 117
predict.sgPLS *(*predict_pls_latent*)*, 117
predict.sPLS *(*predict_pls_latent*)*, 117
predict_cox_pls, 114
predict_pls_latent, 117
print.big_pls_cox, 119
print.big_pls_cox_gd, 120
print.summary.big_pls_cox_fast, 120

select_ncomp *(*component_information*)*, 17
select_ncomp(), *6*, *10*, *113*
sgPLS, *59*
sgPLS::gPLS(), *117, 118*
sgPLS::sgPLS(), *117, 118*
sgPLS::sPLS(), *117, 118*
sim_data, 121
stats::predict(), *114*, *118*
summary.big_pls_cox, 122
summary.big_pls_cox_fast, 122
summary.big_pls_cox_gd, 123
survival::coxph, *62*
survival::coxph(), *9*, *18, 19*
survival::coxph.control, *5*, *7*
survival::coxph.fit, *5*, *7*

tibble::tibble, *22*

Xmicro.censure_compl_imp, 123