

# Package ‘plsRcox’

May 9, 2026

**Version** 1.8.2

**Date** 2026-03-24

**Depends** R (>= 2.4.0)

**Imports** survival, plsRglm, lars, pls, kernlab, mixOmics, risksetROC, survcomp, survAUC, rms

**Suggests** survivalROC, plsdof, testthat (>= 3.0.0)

**Title** Partial Least Squares Regression for Cox Models and Related Techniques

**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 in high dimensional settings <[doi:10.1093/bioinformatics/btu660](https://doi.org/10.1093/bioinformatics/btu660)>, Bastien, P., Bertrand, F., Meyer N., Maumy-Bertrand, M. (2015), Deviance residuals-based sparse PLS and sparse kernel PLS regression for censored data, Bioinformatics, 31(3):397-404. Cross validation criteria were studied in <[doi:10.48550/arXiv.1810.02962](https://doi.org/10.48550/arXiv.1810.02962)>, Bertrand, F., Bastien, Ph. and Maumy-Bertrand, M. (2018), Cross validating extensions of kernel, sparse or regular partial least squares regression models to censored data.

**License** GPL-3

**Encoding** UTF-8

**URL** <https://fbertran.github.io/plsRcox/>,  
<https://github.com/fbertran/plsRcox>

**BugReports** <https://github.com/fbertran/plsRcox/issues>

**Classification/MS** 62N01, 62N02, 62N03, 62N99

**RoxygenNote** 7.3.3

**Config/testthat/edition** 3

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2026-03-24 12:50:02 UTC

## Contents

coxDKpls2DR . . . . .	2
coxDKplsDR . . . . .	7
coxDKsplsDR . . . . .	12
coxpls . . . . .	17
coxpls2 . . . . .	21
coxpls2DR . . . . .	24
coxpls3 . . . . .	28
coxpls3DR . . . . .	32
coxplsDR . . . . .	36
coxspplsDR . . . . .	40
cv.autoplsRcox . . . . .	44
cv.coxDKplsDR . . . . .	48
cv.coxDKsplsDR . . . . .	53
cv.coxpls . . . . .	58
cv.coxplsDR . . . . .	62
cv.coxspplsDR . . . . .	67
cv.larsDR . . . . .	72
cv.plsRcox . . . . .	76
DKplsRcox . . . . .	81
DR_coxph . . . . .	86
larsDR_coxph . . . . .	88
micro.censure . . . . .	92
plsRcox . . . . .	94
predict.coxDRmodel . . . . .	99
predict.plsRcoxmodel . . . . .	100
print.plsRcoxmodel . . . . .	103
print.summary.plsRcoxmodel . . . . .	104
summary.plsRcoxmodel . . . . .	105
Xmicro.censure_compl_imp . . . . .	107

**Index**

**109**

---

coxDKpls2DR

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

---

## Description

This function computes the Direct Kernel PLSR model with the Residuals of a Cox-Model fitted with an intercept as the only explanatory variable as the response and Xplan as explanatory variables. Default behaviour uses the Deviance residuals.

**Usage**

```
coxDKpls2DR(Xplan, ...)

## Default S3 method:
coxDKpls2DR(
  Xplan,
  time,
  time2,
  event,
  type,
  origin,
  typeres = "deviance",
  collapse,
  weighted,
  scaleX = TRUE,
  scaleY = TRUE,
  ncomp = min(7, ncol(Xplan)),
  methodpls = "kernelpls",
  validation = "CV",
  plot = FALSE,
  allres = FALSE,
  kernel = "rbfdot",
  hyperkernel,
  verbose = TRUE,
  ...
)

## S3 method for class 'formula'
coxDKpls2DR(
  Xplan,
  time,
  time2,
  event,
  type,
  origin,
  typeres = "deviance",
  collapse,
  weighted,
  scaleX = TRUE,
  scaleY = TRUE,
  ncomp = min(7, ncol(Xplan)),
  methodpls = "kernelpls",
  validation = "CV",
  plot = FALSE,
  allres = FALSE,
  dataXplan = NULL,
  subset,
  weights,
```

```

    model_frame = FALSE,
    kernel = "rbfdot",
    hyperkernel,
    verbose = TRUE,
    model_matrix = FALSE,
    contrasts.arg = NULL,
    ...
)

```

### Arguments

Xplan	a formula or a matrix with the eXplanatory variables (training) dataset
...	Arguments to be passed on to <code>survival::coxph</code> .
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 <code>collapse=c(1, 1, 1, 2, 3, 3, 4, 4, 4, 4)</code> 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. The number of components to fit is specified with the argument ncomp. If this is not supplied, the maximal number of components is used (taking account of any cross-validation).
methodpls	The multivariate regression method to be used. See <a href="#">mvrCv</a> for details.
validation	character. What kind of (internal) validation to use. If validation = "CV", cross-validation is performed. The number and type of cross-validation segments are specified with the arguments segments and segment.type. See <a href="#">mvrCv</a> for details. If validation = "LOO", leave-one-out cross-validation is performed. It is an error to specify the segments when validation = "LOO" is specified.
plot	Should the survival function be plotted ?
allres	FALSE to return only the Cox model and TRUE for additional results. See details. Defaults to FALSE.
kernel	<p>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 <a href="#">kernels</a>). The kernlab package provides the most popular kernel functions which can be used by setting the kernel parameter to the following strings:</p> <p><b>list("rbfdot")</b> Radial Basis kernel "Gaussian"  <b>list("polydot")</b> Polynomial kernel  <b>list("vanilladot")</b> Linear kernel  <b>list("tanhdot")</b> Hyperbolic tangent kernel  <b>list("laplacedot")</b> Laplacian kernel  <b>list("besseldot")</b> Bessel kernel  <b>list("anovadot")</b> ANOVA RBF kernel  <b>list("splinedot")</b> Spline kernel</p>
hyperkernel	<p>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 :</p> <ul style="list-style-type: none"> <li>• sigma, inverse kernel width for the Radial Basis kernel function "rbfdot" and the Laplacian kernel "laplacedot".</li> <li>• degree, scale, offset for the Polynomial kernel "polydot".</li> <li>• scale, offset for the Hyperbolic tangent kernel function "tanhdot".</li> <li>• sigma, order, degree for the Bessel kernel "besseldot".</li> <li>• sigma, degree for the ANOVA kernel "anovadot".</li> </ul> <p>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.</p>
verbose	Should some details be displayed ?
dataXplan	an optional data frame, list or environment (or object coercible by <a href="#">as.data.frame</a> 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 coxDKpls2DR 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 PLSR model. `allres=TRUE` is useful for evaluating model prediction accuracy on a test sample.

### Value

If `allres=FALSE` :

`cox_DKpls2DR` Final Cox-model.

If `allres=TRUE` :

`tt_DKpls2DR` PLSR components.

`cox_DKpls2DR` Final Cox-model.

`DKpls2DR_mod` The PLSR model.

### 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.

### See Also

[coxph](#), [pls](#)

**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_DKpls2DR_fit=coxDKpls2DR(X_train_micro,Y_train_micro,C_train_micro,ncomp=6,validation="CV"))

#Fixing sigma to compare with pls2DR on Gram matrix; should be identical
(cox_DKpls2DR_fit=coxDKpls2DR(X_train_micro,Y_train_micro,C_train_micro,ncomp=6,
validation="CV",hyperkernel=list(sigma=0.01292786)))

X_train_micro_kern <- kernlab::kernelMatrix(kernlab::rbfdot(sigma=0.01292786),scale(X_train_micro))
(cox_DKpls2DR_fit2=coxpls2DR(~X_train_micro_kern,Y_train_micro,C_train_micro,ncomp=6,
validation="CV",scaleX=FALSE))

(cox_DKpls2DR_fit=coxDKpls2DR(X_train_micro,Y_train_micro,C_train_micro,ncomp=6,
validation="CV",kernel="laplacedot",hyperkernel=list(sigma=0.01292786)))

X_train_micro_kern <- kernlab::kernelMatrix(kernlab::laplacedot(sigma=0.01292786),
scale(X_train_micro))
(cox_DKpls2DR_fit2=coxpls2DR(~X_train_micro_kern,Y_train_micro,C_train_micro,ncomp=6,
validation="CV",scaleX=FALSE))

(cox_DKpls2DR_fit=coxDKpls2DR(~X_train_micro,Y_train_micro,C_train_micro,ncomp=6,validation="CV"))
(cox_DKpls2DR_fit=coxDKpls2DR(~.,Y_train_micro,C_train_micro,ncomp=6,validation="CV",
dataXplan=X_train_micro_df))

(cox_DKpls2DR_fit=coxDKpls2DR(X_train_micro,Y_train_micro,C_train_micro,ncomp=6,
validation="CV",allres=TRUE))
(cox_DKpls2DR_fit=coxDKpls2DR(~X_train_micro,Y_train_micro,C_train_micro,ncomp=6,
validation="CV",allres=TRUE))
(cox_DKpls2DR_fit=coxDKpls2DR(~.,Y_train_micro,C_train_micro,ncomp=6,validation="CV",
allres=TRUE,dataXplan=X_train_micro_df))

rm(X_train_micro,Y_train_micro,C_train_micro,cox_DKpls2DR_fit)

```

---

coxDKplsDR

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


---

**Description**

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

- as the response: the Residuals of a Cox-Model fitted with no covariate

- as explanatory variables: a Kernel transform of Xplan.

It uses the package kernlab to compute the Kernel transforms of Xplan, then the package mixOmics to perform PLSR fit.

### Usage

```
coxDKplsDR(Xplan, ...)

## Default S3 method:
coxDKplsDR(
  Xplan,
  time,
  time2,
  event,
  type,
  origin,
  typeres = "deviance",
  collapse,
  weighted,
  scaleX = TRUE,
  scaleY = TRUE,
  ncomp = min(7, ncol(Xplan)),
  modepls = "regression",
  plot = FALSE,
  allres = FALSE,
  kernel = "rbfdot",
  hyperkernel,
  verbose = TRUE,
  ...
)

## S3 method for class 'formula'
coxDKplsDR(
  Xplan,
  time,
  time2,
  event,
  type,
  origin,
  typeres = "deviance",
  collapse,
  weighted,
  scaleX = TRUE,
  scaleY = TRUE,
  ncomp = min(7, ncol(Xplan)),
  modepls = "regression",
  plot = FALSE,
  allres = FALSE,
```

```

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

```

### Arguments

Xplan	a formula or a matrix with the eXplanatory variables (training) dataset
...	Arguments to be passed on to <code>survival::coxph</code> .
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 <code>collapse=c(1, 1, 1, 2, 3, 3, 4, 4, 4, 4)</code> 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. The number of components to fit is specified with the argument ncomp. If this is not supplied, the maximal number of components is used.
modepls	character string. What type of algorithm to use, (partially) matching one of "regression", "canonical", "invariant" or "classic". See <a href="#">pls</a> for details
plot	Should the survival function be plotted ?
allres	FALSE to return only the Cox model and TRUE for additional results. See details. Defaults to FALSE.
kernel	<p>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 <a href="#">kernels</a>). The kernLab package provides the most popular kernel functions which can be used by setting the kernel parameter to the following strings:</p> <p><b>list("rbfdot")</b> Radial Basis kernel "Gaussian"  <b>list("polydot")</b> Polynomial kernel  <b>list("vanilladot")</b> Linear kernel  <b>list("tanhdot")</b> Hyperbolic tangent kernel  <b>list("laplacedot")</b> Laplacian kernel  <b>list("besseldot")</b> Bessel kernel  <b>list("anovadot")</b> ANOVA RBF kernel  <b>list("splinedot")</b> Spline kernel</p>
hyperkernel	<p>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 :</p> <ul style="list-style-type: none"> <li>• sigma, inverse kernel width for the Radial Basis kernel function "rbfdot" and the Laplacian kernel "laplacedot".</li> <li>• degree, scale, offset for the Polynomial kernel "polydot".</li> <li>• scale, offset for the Hyperbolic tangent kernel function "tanhdot".</li> <li>• sigma, order, degree for the Bessel kernel "besseldot".</li> <li>• sigma, degree for the ANOVA kernel "anovadot".</li> </ul> <p>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.</p>
verbose	Should some details be displayed ?
dataXplan	an optional data frame, list or environment (or object coercible by <a href="#">as.data.frame</a> 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 coxDKplsDR 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 PLSR model. allres=TRUE is useful for evaluating model prediction accuracy on a test sample.

### Value

If allres=FALSE :

cox\_DKplsDR      Final Cox-model.

If allres=TRUE :

tt\_DKplsDR      PLSR components.

cox\_DKplsDR      Final Cox-model.

DKplsDR\_mod      The PLSR model.

### 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.

### See Also

[coxph](#), [pls](#)

### 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_DKplsDR_fit=coxDKplsDR(X_train_micro,Y_train_micro,C_train_micro,ncomp=6))

#Fixing sigma to compare with plsDR on Gram matrix; should be identical
(cox_DKplsDR_fit=coxDKplsDR(X_train_micro,Y_train_micro,C_train_micro,ncomp=6,
hyperkernel=list(sigma=0.01292786)))

X_train_micro_kern <- kernlab::kernelMatrix(kernlab::rbfdot(sigma=0.01292786),
scale(X_train_micro))
(cox_DKplsDR_fit2=coxplsDR(~X_train_micro_kern,Y_train_micro,C_train_micro,ncomp=6,scaleX=FALSE))

(cox_DKplsDR_fit=coxDKplsDR(X_train_micro,Y_train_micro,C_train_micro,ncomp=6,
kernel="laplacedot",hyperkernel=list(sigma=0.01292786)))

X_train_micro_kern <- kernlab::kernelMatrix(kernlab::laplacedot(sigma=0.01292786),
scale(X_train_micro))
(cox_DKplsDR_fit2=coxplsDR(~X_train_micro_kern,Y_train_micro,C_train_micro,ncomp=6,scaleX=FALSE))

(cox_DKplsDR_fit=coxDKplsDR(~X_train_micro,Y_train_micro,C_train_micro,ncomp=6))
(cox_DKplsDR_fit=coxDKplsDR(~.,Y_train_micro,C_train_micro,ncomp=6,dataXplan=X_train_micro_df))

(cox_DKplsDR_fit=coxDKplsDR(X_train_micro,Y_train_micro,C_train_micro,ncomp=6,allres=TRUE))
(cox_DKplsDR_fit=coxDKplsDR(~X_train_micro,Y_train_micro,C_train_micro,ncomp=6,allres=TRUE))
(cox_DKplsDR_fit=coxDKplsDR(~.,Y_train_micro,C_train_micro,ncomp=6,allres=TRUE,
dataXplan=X_train_micro_df))

rm(X_train_micro,Y_train_micro,C_train_micro,cox_DKplsDR_fit)

```

---

coxDKsplDR

*Fitting a Direct Kernel sPLSR model on the (Deviance) Residuals*


---

## Description

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

- as the response: the Residuals of a Cox-Model fitted with no covariate
- as explanatory variables: a Kernel transform of Xplan.

It uses the package kernlab to compute the Kernel transforms of Xplan, the package spls to perform the first step in SPLSR then mixOmics to perform PLSR step fit.

## Usage

```
coxDKsplDR(Xplan, ...)
```

```
## Default S3 method:
```

```
coxDKsplDR(  
  Xplan,  
  time,  
  time2,  
  event,  
  type,  
  origin,  
  typeres = "deviance",  
  collapse,  
  weighted,  
  scaleX = TRUE,  
  scaleY = TRUE,  
  ncomp = min(7, ncol(Xplan)),  
  modepls = "regression",  
  plot = FALSE,  
  allres = FALSE,  
  eta,  
  trace = FALSE,  
  kernel = "rbfdot",  
  hyperkernel,  
  verbose = TRUE,  
  ...  
)  
  
## S3 method for class 'formula'  
coxDKsplDR(  
  Xplan,  
  time,  
  time2,  
  event,  
  type,  
  origin,  
  typeres = "deviance",  
  collapse,  
  weighted,  
  scaleX = TRUE,  
  scaleY = TRUE,  
  ncomp = min(7, ncol(Xplan)),  
  modepls = "regression",  
  plot = FALSE,  
  allres = FALSE,  
  dataXplan = NULL,  
  subset,  
  weights,  
  model_frame = FALSE,  
  eta,  
  trace = FALSE,  
  kernel = "rbfdot",
```

```

hyperkernel,
verbose = TRUE,
model_matrix = FALSE,
contrasts.arg = NULL,
...
)

```

## Arguments

<code>Xplan</code>	a formula or a matrix with the eXplanatory variables (training) dataset
<code>...</code>	Arguments to be passed on to <code>survival::coxph</code> .
<code>time</code>	for right censored data, this is the follow up time. For interval data, the first argument is the starting time for the interval.
<code>time2</code>	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.
<code>event</code>	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, ( <code>start</code> , <code>end</code> ]. For counting process data, <code>event</code> indicates whether an event occurred at the end of the interval.
<code>type</code>	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 <code>time2</code> argument is absent or present, respectively.
<code>origin</code>	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.
<code>typeres</code>	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.
<code>collapse</code>	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 <code>collapse=c(1, 1, 1, 2, 3, 3, 4, 4, 4, 4)</code> could be used to obtain per subject rather than per observation residuals.
<code>weighted</code>	if TRUE and the model was fit with case weights, then the weighted residuals are returned.
<code>scaleX</code>	Should the <code>Xplan</code> columns be standardized ?
<code>scaleY</code>	Should the <code>time</code> values be standardized ?
<code>ncomp</code>	The number of components to include in the model. The number of components to fit is specified with the argument <code>ncomp</code> . If this is not supplied, the maximal number of components is used.

modepls	character string. What type of algorithm to use, (partially) matching one of "regression", "canonical", "invariant" or "classic". See <a href="#">pls</a> for details
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.
eta	Thresholding parameter. eta should be between 0 and 1.
trace	Print out the progress of variable selection?
kernel	<p>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 <a href="#">kernels</a>). The kernlab package provides the most popular kernel functions which can be used by setting the kernel parameter to the following strings:</p> <p><b>list("rbfdot")</b> Radial Basis kernel "Gaussian"  <b>list("polydot")</b> Polynomial kernel  <b>list("vanilladot")</b> Linear kernel  <b>list("tanhdot")</b> Hyperbolic tangent kernel  <b>list("laplacedot")</b> Laplacian kernel  <b>list("besseldot")</b> Bessel kernel  <b>list("anovadot")</b> ANOVA RBF kernel  <b>list("splinedot")</b> Spline kernel</p>
hyperkernel	<p>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 :</p> <ul style="list-style-type: none"> <li>• sigma, inverse kernel width for the Radial Basis kernel function "rbfdot" and the Laplacian kernel "laplacedot".</li> <li>• degree, scale, offset for the Polynomial kernel "polydot".</li> <li>• scale, offset for the Hyperbolic tangent kernel function "tanhdot".</li> <li>• sigma, order, degree for the Bessel kernel "besseldot".</li> <li>• sigma, degree for the ANOVA kernel "anovadot".</li> </ul> <p>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.</p>
verbose	Should some details be displayed ?
dataXplan	an optional data frame, list or environment (or object coercible by <a href="#">as.data.frame</a> 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 coxDKsplSDR 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 sPLS components, the final Cox-model and the sPLSR model. `allres=TRUE` is useful for evaluating model prediction accuracy on a test sample.

### Value

If `allres=FALSE` :

`cox_DKsplSDR` Final Cox-model.

If `allres=TRUE` :

`tt_DKsplSDR` sPLSR components.

`cox_DKsplSDR` Final Cox-model.

`DKsplSDR_mod` The sPLSR model.

### 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.

### See Also

[coxph](#), [pls](#)

### 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_DKsplsDR_fit=coxDKsplsDR(X_train_micro,Y_train_micro,C_train_micro,ncomp=6,
validation="CV",eta=.5))
(cox_DKsplsDR_fit=coxDKsplsDR(~X_train_micro,Y_train_micro,C_train_micro,ncomp=6,
validation="CV",eta=.5))
(cox_DKsplsDR_fit=coxDKsplsDR(~.,Y_train_micro,C_train_micro,ncomp=6,
validation="CV",dataXplan=data.frame(X_train_micro),eta=.5))

(cox_DKsplsDR_fit=coxDKsplsDR(X_train_micro,Y_train_micro,C_train_micro,ncomp=6,
validation="CV",allres=TRUE,eta=.5))
(cox_DKsplsDR_fit=coxDKsplsDR(~X_train_micro,Y_train_micro,C_train_micro,ncomp=6,
validation="CV",allres=TRUE,eta=.5))
(cox_DKsplsDR_fit=coxDKsplsDR(~.,Y_train_micro,C_train_micro,ncomp=6,
validation="CV",allres=TRUE,dataXplan=data.frame(X_train_micro),eta=.5))

rm(X_train_micro,Y_train_micro,C_train_micro,cox_DKsplsDR_fit)

```

---

coxpls

*Fitting a Cox-Model on 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 `mixOmics` to perform PLSR fit.

## Usage

```

coxpls(Xplan, ...)

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

```

```

modepls = "regression",
plot = FALSE,
allres = FALSE,
...
)

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

```

### Arguments

Xplan	a formula or a matrix with the eXplanatory variables (training) dataset
...	Arguments to be passed on to <code>survival::coxph</code> .
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, ( <code>start</code> , <code>end</code> ]. 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. If 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", "invariant" or "classic". See <a href="#">pls</a> for details
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 <a href="#">as.data.frame</a> 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 PLSR model. allres=TRUE is useful for evaluating model prediction accuracy on a test sample.

**Value**

If `allres=FALSE` :

`cox_pls`            Final Cox-model.

If `allres=TRUE` :

`tt_pls`            PLSR components.

`cox_pls`            Final Cox-model.

`pls_mod`            The PLSR model.

**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.

**See Also**

[coxph](#), [pls](#)

**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_pls_fit=coxpls(X_train_micro,Y_train_micro,C_train_micro,ncomp=6))
(cox_pls_fit=coxpls(~X_train_micro,Y_train_micro,C_train_micro,ncomp=6))
(cox_pls_fit=coxpls(~.,Y_train_micro,C_train_micro,ncomp=6,dataXplan=X_train_micro_df))

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

---

`coxpls2`*Fitting a Cox-Model on PLSR components*

---

**Description**

This function computes the the Cox-Model with PLSR components as the explanatory variables. It uses the package `pls`.

**Usage**

```
coxpls2(Xplan, ...)  
  
## Default S3 method:  
coxpls2(  
  Xplan,  
  time,  
  time2,  
  event,  
  type,  
  origin,  
  typeres = "deviance",  
  collapse,  
  weighted,  
  scaleX = TRUE,  
  scaleY = TRUE,  
  ncomp = min(7, ncol(Xplan)),  
  methodpls = "kernelpls",  
  validation = "CV",  
  plot = FALSE,  
  allres = FALSE,  
  ...  
)  
  
## S3 method for class 'formula'  
coxpls2(  
  Xplan,  
  time,  
  time2,  
  event,  
  type,  
  origin,  
  typeres = "deviance",  
  collapse,  
  weighted,  
  scaleX = TRUE,  
  scaleY = TRUE,  
  ncomp = min(7, ncol(Xplan)),
```

```

methodpls = "kernelpls",
validation = "CV",
plot = FALSE,
allres = FALSE,
dataXplan = NULL,
subset,
weights,
model_frame = FALSE,
model_matrix = FALSE,
contrasts.arg = NULL,
...
)

```

### Arguments

Xplan	a formula or a matrix with the eXplanatory variables (training) dataset
...	Arguments to be passed on to <code>survival::coxph</code> .
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 <code>collapse=c(1, 1, 1, 2, 3, 3, 4, 4, 4, 4)</code> 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. The number of components to fit is specified with the argument ncomp. If this is not supplied, the maximal number of components is used (taking account of any cross-validation).
methodpls	The multivariate regression method to be used. See <a href="#">mvrCv</a> for details.
validation	character. What kind of (internal) validation to use. If validation = "CV", cross-validation is performed. The number and type of cross-validation segments are specified with the arguments segments and segment.type. See <a href="#">mvrCv</a> for details. If validation = "LOO", leave-one-out cross-validation is performed. It is an error to specify the segments when validation = "LOO" is specified.
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 <a href="#">as.data.frame</a> 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 coxpls2 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 PLSR model. allres=TRUE is useful for evaluating model prediction accuracy on a test sample.

### Value

If allres=FALSE :

cox\_pls            Final Cox-model.

If allres=TRUE :

tt\_pls            PLSR components.

cox\_pls            Final Cox-model.

pls\_mod            The PLSR model.

**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.

**See Also**

[coxph](#), [pls](#)

**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_pls_fit=coxpls2(X_train_micro,Y_train_micro,C_train_micro,ncomp=6,validation="CV"))
(cox_pls_fit=coxpls2(~X_train_micro,Y_train_micro,C_train_micro,ncomp=6,validation="CV"))
(cox_pls_fit=coxpls2(~.,Y_train_micro,C_train_micro,ncomp=6,validation="CV",
dataXplan=X_train_micro_df))

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

---

 coxpls2DR

*Fitting a PLSR model on the (Deviance) Residuals*


---

**Description**

This function computes the PLSR model with the Residuals of a Cox-Model fitted with an intercept as the only explanatory variable as the response and Xplan as explanatory variables. Default behaviour uses the Deviance residuals. It uses the package pls.

**Usage**

```
coxpls2DR(Xplan, ...)

## Default S3 method:
coxpls2DR(
  Xplan,
  time,
  time2,
  event,
  type,
  origin,
  typeres = "deviance",
  collapse,
  weighted,
  scaleX = TRUE,
  scaleY = TRUE,
  ncomp = min(7, ncol(Xplan)),
  methodpls = "kernelpls",
  validation = "CV",
  plot = FALSE,
  allres = FALSE,
  ...
)

## S3 method for class 'formula'
coxpls2DR(
  Xplan,
  time,
  time2,
  event,
  type,
  origin,
  typeres = "deviance",
  collapse,
  weighted,
  scaleX = TRUE,
  scaleY = TRUE,
  ncomp = min(7, ncol(Xplan)),
  methodpls = "kernelpls",
  validation = "CV",
  plot = FALSE,
  allres = FALSE,
  dataXplan = NULL,
  subset,
  weights,
  model_frame = FALSE,
  model_matrix = FALSE,
  contrasts.arg = NULL,
```

...  
)

### Arguments

Xplan	a formula or a matrix with the eXplanatory variables (training) dataset
...	Arguments to be passed on to <code>survival::coxph</code> .
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 <code>collapse=c(1, 1, 1, 2, 3, 3, 4, 4, 4, 4)</code> 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. The number of components to fit is specified with the argument ncomp. If this is not supplied, the maximal number of components is used (taking account of any cross-validation).
methodpls	The multivariate regression method to be used. See <code>mvrCv</code> for details.
validation	character. What kind of (internal) validation to use. If <code>validation = "CV"</code> , cross-validation is performed. The number and type of cross-validation segments are specified with the arguments <code>segments</code> and <code>segment.type</code> . See

[mvrCv](#) for details. If `validation = "LOO"`, leave-one-out cross-validation is performed. It is an error to specify the segments when `validation = "LOO"` is specified.

<code>plot</code>	Should the survival function be plotted ?)
<code>allres</code>	FALSE to return only the Cox model and TRUE for additionnal results. See details. Defaults to FALSE.
<code>dataXplan</code>	an optional data frame, list or environment (or object coercible by <a href="#">as.data.frame</a> to a data frame) containing the variables in the model. If not found in <code>dataXplan</code> , the variables are taken from <code>environment(Xplan)</code> , typically the environment from which <code>coxpls2DR</code> is called.
<code>subset</code>	an optional vector specifying a subset of observations to be used in the fitting process.
<code>weights</code>	an optional vector of 'prior weights' to be used in the fitting process. Should be NULL or a numeric vector.
<code>model_frame</code>	If TRUE, the model frame is returned.
<code>model_matrix</code>	If TRUE, the model matrix is returned.
<code>contrasts.arg</code>	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 PLSR model. `allres=TRUE` is useful for evaluating model prediction accuracy on a test sample.

### Value

If `allres=FALSE` :

`cox_pls2DR` Final Cox-model.

If `allres=TRUE` :

`tt_pls2DR` PLSR components.

`cox_pls2DR` Final Cox-model.

`pls2DR_mod` The PLSR model.

### Author(s)

Frédéric Bertrand  
 <[frederic.bertrand@lecnam.net](mailto: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.

## See Also

[coxph](#), [pls](#)

## 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_pls2DR_fit=coxpls2DR(X_train_micro,Y_train_micro,C_train_micro,ncomp=6,validation="none"))
(cox_pls2DR_fit2=coxpls2DR(~X_train_micro,Y_train_micro,C_train_micro,ncomp=6,validation="none"))
(cox_pls2DR_fit3=coxpls2DR(~.,Y_train_micro,C_train_micro,ncomp=6,validation="none",
dataXplan=X_train_micro_df))

rm(X_train_micro,Y_train_micro,C_train_micro,cox_pls2DR_fit,cox_pls2DR_fit2,cox_pls2DR_fit3)
```

---

coxpls3

*Fitting a Cox-Model on PLSR components*

---

## Description

This function computes the the Cox-Model with PLSR components as the explanatory variables. It uses the package `plsRglm`.

## Usage

```
coxpls3(Xplan, ...)

## Default S3 method:
coxpls3(
  Xplan,
  time,
  time2,
  event,
```

```

    type,
    origin,
    typeres = "deviance",
    collapse,
    weighted,
    scaleX = TRUE,
    scaleY = TRUE,
    nt = min(7, ncol(Xplan)),
    typeVC = "none",
    plot = FALSE,
    allres = FALSE,
    sparse = FALSE,
    sparseStop = TRUE,
    ...
)

## S3 method for class 'formula'
coxpls3(
  Xplan,
  time,
  time2,
  event,
  type,
  origin,
  typeres = "deviance",
  collapse,
  weighted,
  scaleX = TRUE,
  scaleY = TRUE,
  nt = min(7, ncol(Xplan)),
  typeVC = "none",
  plot = FALSE,
  allres = FALSE,
  dataXplan = NULL,
  subset,
  weights,
  model_frame = FALSE,
  sparse = FALSE,
  sparseStop = TRUE,
  model_matrix = FALSE,
  contrasts.arg = NULL,
  ...
)

```

### Arguments

Xplan            a formula or a matrix with the eXplanatory variables (training) dataset  
 ...             Arguments to be passed on to `survival::coxph` and to `plsRglm::PLS_lm`.

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 ?
nt	Number of PLSR components to fit.
typeVC	type of leave one out crossed validation. Several procedures are available and may be forced. <b>list("none")</b> no crossed validation <b>list("standard")</b> as in SIMCA for datasets without missing values and with all values predicted as those with missing values for datasets with any missing values <b>list("missingdata")</b> all values predicted as those with missing values for datasets with any missing values <b>list("adaptative")</b> predict a response value for an x with any missing value as those with missing values and for an x without any missing value as those without missing values.
plot	Should the survival function be plotted ?)

<code>allres</code>	FALSE to return only the Cox model and TRUE for additionnal results. See details. Defaults to FALSE.
<code>sparse</code>	should the coefficients of non-significant predictors ( <code>&lt;alpha.pvals.expli</code> ) be set to 0
<code>sparseStop</code>	should component extraction stop when no significant predictors ( <code>&lt;alpha.pvals.expli</code> ) are found
<code>dataXplan</code>	an optional data frame, list or environment (or object coercible by <code>as.data.frame</code> to a data frame) containing the variables in the model. If not found in <code>dataXplan</code> , the variables are taken from <code>environment(Xplan)</code> , typically the environment from which <code>coxpls3</code> is called.
<code>subset</code>	an optional vector specifying a subset of observations to be used in the fitting process.
<code>weights</code>	an optional vector of 'prior weights' to be used in the fitting process. Should be NULL or a numeric vector.
<code>model_frame</code>	If TRUE, the model frame is returned.
<code>model_matrix</code>	If TRUE, the model matrix is returned.
<code>contrasts.arg</code>	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 PLSR model. `allres=TRUE` is useful for evaluating model prediction accuracy on a test sample.

### Value

If `allres=FALSE` :

`cox_pls3` Final Cox-model.

If `allres=TRUE` :

`tt_pls3` PLSR components.

`cox_pls3` Final Cox-model.

`pls3_mod` The PLSR model.

### 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.

## See Also

[coxph](#), [PLS\\_lm](#)

## 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_pls3_fit <- coxpls3(X_train_micro,Y_train_micro,C_train_micro,nt=7,typeVC="none"))
(cox_pls3_fit2 <- coxpls3(~X_train_micro,Y_train_micro,C_train_micro,nt=7,typeVC="none"))
(cox_pls3_fit3 <- coxpls3(~.,Y_train_micro,C_train_micro,nt=7,typeVC="none",data=X_train_micro_df))
(cox_pls3_fit4 <- coxpls3(~.,Y_train_micro,C_train_micro,nt=7,typeVC="none",
data=X_train_micro_df,sparse=TRUE))
(cox_pls3_fit5 <- coxpls3(~.,Y_train_micro,C_train_micro,nt=7,typeVC="none",
data=X_train_micro_df,sparse=FALSE,sparseStop=TRUE))

rm(X_train_micro,Y_train_micro,C_train_micro,cox_pls3_fit,cox_pls3_fit2,
cox_pls3_fit3,cox_pls3_fit4,cox_pls3_fit5)
```

---

coxpls3DR

*Fitting a PLSR model on the (Deviance) Residuals*

---

## Description

This function computes the PLSR model with the Residuals of a Cox-Model fitted with an intercept as the only explanatory variable as the response and Xplan as explanatory variables. Default behaviour uses the Deviance residuals. It uses the package `plsRglm`.

## Usage

```
coxpls3DR(Xplan, ...)

## Default S3 method:
```

```
coxpls3DR(  
  Xplan,  
  time,  
  time2,  
  event,  
  type,  
  origin,  
  typeres = "deviance",  
  collapse,  
  weighted,  
  scaleX = TRUE,  
  scaleY = TRUE,  
  nt = min(7, ncol(Xplan)),  
  typeVC = "none",  
  plot = FALSE,  
  allres = FALSE,  
  sparse = FALSE,  
  sparseStop = TRUE,  
  ...  
)  
  
## S3 method for class 'formula'  
coxpls3DR(  
  Xplan,  
  time,  
  time2,  
  event,  
  type,  
  origin,  
  typeres = "deviance",  
  collapse,  
  weighted,  
  scaleX = TRUE,  
  scaleY = TRUE,  
  nt = min(7, ncol(Xplan)),  
  typeVC = "none",  
  plot = FALSE,  
  allres = FALSE,  
  dataXplan = NULL,  
  subset,  
  weights,  
  model_frame = FALSE,  
  sparse = FALSE,  
  sparseStop = TRUE,  
  model_matrix = FALSE,  
  contrasts.arg = NULL,  
  ...  
)
```

**Arguments**

Xplan	a formula or a matrix with the eXplanatory variables (training) dataset
...	Arguments to be passed on to <code>survival::coxph</code> and to <code>plsRglm::PLS_lm</code> .
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 <code>collapse=c(1, 1, 1, 2, 3, 3, 4, 4, 4, 4)</code> 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 ?
nt	Number of PLSR components to fit.
typeVC	type of leave one out crossed validation. Several procedures are available and may be forced. <b>list("none")</b> no crossed validation <b>list("standard")</b> as in SIMCA for datasets without missing values and with all values predicted as those with missing values for datasets with any missing values <b>list("missingdata")</b> all values predicted as those with missing values for datasets with any missing values

	<b>list("adaptative")</b> predict a response value for an x with any missing value as those with missing values and for an x without any missing value as those without missing values.
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.
sparse	should the coefficients of non-significant predictors (<alpha.pvals.expli) be set to 0
sparseStop	should component extraction stop when no significant predictors (<alpha.pvals.expli) are found
dataXplan	an optional data frame, list or environment (or object coercible by <a href="#">as.data.frame</a> 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 coxpls3DR 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 PLSR model. allres=TRUE is useful for evaluating model prediction accuracy on a test sample.

### Value

If allres=FALSE :

cox\_pls3DR      Final Cox-model.

If allres=TRUE :

tt\_pls3DR      PLSR components.

cox\_pls3DR      Final Cox-model.

pls3DR\_mod      The PLSR model.

### 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.

## See Also

[coxph](#), [PLS\\_lm](#)

## 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_pls3DR_fit <- coxpls3DR(X_train_micro,Y_train_micro,C_train_micro,nt=7))
(cox_pls3DR_fit2 <- coxpls3DR(~X_train_micro,Y_train_micro,C_train_micro,nt=7))
(cox_pls3DR_fit3 <- coxpls3DR(~.,Y_train_micro,C_train_micro,nt=7,dataXplan=X_train_micro_df))
(cox_pls3DR_fit4 <- coxpls3DR(~.,Y_train_micro,C_train_micro,nt=7,typeVC="none",
data=X_train_micro_df,sparse=TRUE))
(cox_pls3DR_fit5 <- coxpls3DR(~.,Y_train_micro,C_train_micro,nt=7,typeVC="none",
data=X_train_micro_df,sparse=TRUE,sparseStop=FALSE))

rm(X_train_micro,Y_train_micro,C_train_micro,cox_pls3DR_fit,cox_pls3DR_fit2,
cox_pls3DR_fit3,cox_pls3DR_fit4,cox_pls3DR_fit5)
```

---

coxplsDR

*Fitting a PLSR model on the (Deviance) Residuals*

---

## Description

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

- as the response: the Residuals of a Cox-Model fitted with no covariate
- as explanatory variables: Xplan.

It uses the package `mixOmics` to perform PLSR fit.

**Usage**

```
coxplsDR(Xplan, ...)

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

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

**Arguments**

<code>Xplan</code>	a formula or a matrix with the eXplanatory variables (training) dataset
<code>...</code>	Arguments to be passed on to <code>survival::coxph</code> .
<code>time</code>	for right censored data, this is the follow up time. For interval data, the first argument is the starting time for the interval.
<code>time2</code>	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.
<code>event</code>	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, ( <code>start</code> , <code>end</code> ]. For counting process data, event indicates whether an event occurred at the end of the interval.
<code>type</code>	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 <code>time2</code> argument is absent or present, respectively.
<code>origin</code>	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.
<code>typeres</code>	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.
<code>collapse</code>	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 <code>collapse=c(1, 1, 1, 2, 3, 3, 4, 4, 4, 4)</code> could be used to obtain per subject rather than per observation residuals.
<code>weighted</code>	if TRUE and the model was fit with case weights, then the weighted residuals are returned.
<code>scaleX</code>	Should the <code>Xplan</code> columns be standardized ?
<code>scaleY</code>	Should the time values be standardized ?
<code>ncomp</code>	The number of components to include in the model. The number of components to fit is specified with the argument <code>ncomp</code> . If this is not supplied, the maximal number of components is used.
<code>modepls</code>	character string. What type of algorithm to use, (partially) matching one of "regression", "canonical", "invariant" or "classic". See <a href="#">pls</a> for details
<code>plot</code>	Should the survival function be plotted ?
<code>allres</code>	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 <code>as.data.frame</code> to a data frame) containing the variables in the model. If not found in dataXplan, the variables are taken from <code>environment(Xplan)</code> , typically the environment from which <code>coxplsDR</code> 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 PLSR model. `allres=TRUE` is useful for evaluating model prediction accuracy on a test sample.

### Value

If `allres=FALSE` :

`cox_plsDR` Final Cox-model.

If `allres=TRUE` :

`tt_plsDR` PLSR components.

`cox_plsDR` Final Cox-model.

`plsDR_mod` The PLSR model.

### 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.

**See Also**[coxph](#), [pls](#)**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_plsDR_fit=coxspplsDR(X_train_micro,Y_train_micro,C_train_micro,ncomp=6))
(cox_plsDR_fit2=coxspplsDR(~X_train_micro,Y_train_micro,C_train_micro,ncomp=6))
(cox_plsDR_fit3=coxspplsDR(~.,Y_train_micro,C_train_micro,ncomp=6,dataXplan=X_train_micro_df))

rm(X_train_micro,Y_train_micro,C_train_micro,cox_plsDR_fit,cox_plsDR_fit2,cox_plsDR_fit3)

```

---

coxspplsDR

*Fitting a sPLSR model on the (Deviance) Residuals*


---

**Description**

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

- as the response: the Residuals of a Cox-Model fitted with no covariate
- as explanatory variables: Xplan.

It uses the package `spls` to perform the first step in SPLSR then `mixOmics` to perform PLSR step fit.

**Usage**

```

coxspplsDR(Xplan, ...)

## Default S3 method:
coxspplsDR(
  Xplan,
  time,
  time2,
  event,
  type,
  origin,
  typeres = "deviance",
  collapse,
  weighted,

```

```

    scaleX = TRUE,
    scaleY = TRUE,
    ncomp = min(7, ncol(Xplan)),
    modepls = "regression",
    plot = FALSE,
    allres = FALSE,
    eta = 0.5,
    trace = FALSE,
    ...
)

## S3 method for class 'formula'
coxplsDR(
  Xplan,
  time,
  time2,
  event,
  type,
  origin,
  typeres = "deviance",
  collapse,
  weighted,
  scaleX = TRUE,
  scaleY = TRUE,
  ncomp = min(7, ncol(Xplan)),
  modepls = "regression",
  plot = FALSE,
  allres = FALSE,
  dataXplan = NULL,
  subset,
  weights,
  model_frame = FALSE,
  eta = 0.5,
  trace = FALSE,
  model_matrix = FALSE,
  contrasts.arg = NULL,
  ...
)

```

### Arguments

<code>Xplan</code>	a formula or a matrix with the eXplanatory variables (training) dataset
<code>...</code>	Arguments to be passed on to <code>survival::coxph</code> .
<code>time</code>	for right censored data, this is the follow up time. For interval data, the first argument is the starting time for the interval.
<code>time2</code>	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. The number of components to fit is specified with the argument ncomp. If this is not supplied, the maximal number of components is used.
modepls	character string. What type of algorithm to use, (partially) matching one of "regression", "canonical", "invariant" or "classic". See <a href="#">pls</a> for details
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.
eta	Thresholding parameter. eta should be between 0 and 1.
trace	Print out the progress of variable selection?
dataXplan	an optional data frame, list or environment (or object coercible by <a href="#">as.data.frame</a> 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 coxsp1sDR 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.

<code>model_frame</code>	If TRUE, the model frame is returned.
<code>model_matrix</code>	If TRUE, the model matrix is returned.
<code>contrasts.arg</code>	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 sPLS components, the final Cox-model and the sPLSR model. `allres=TRUE` is useful for evaluating model prediction accuracy on a test sample.

### Value

If `allres=FALSE` :

`cox_splsDR` Final Cox-model.

If `allres=TRUE` :

`tt_splsDR` sPLSR components.

`cox_splsDR` Final Cox-model.

`splsDR_mod` The sPLSR model.

### 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.

### See Also

[coxph](#), [pls](#)

### 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_splsDR_fit=coxspplsDR(X_train_micro,Y_train_micro,C_train_micro,ncomp=6,eta=.5))
(cox_splsDR_fit2=coxspplsDR(~X_train_micro,Y_train_micro,C_train_micro,ncomp=6,eta=.5,trace=TRUE))
(cox_splsDR_fit3=coxspplsDR(~.,Y_train_micro,C_train_micro,ncomp=6,
dataXplan=X_train_micro_df,eta=.5))

rm(X_train_micro,Y_train_micro,C_train_micro,cox_splsDR_fit,cox_splsDR_fit2,cox_splsDR_fit3)

```

---

cv.autoplsRcox

*Cross-validating an autoplsRcox-Model*


---

### Description

This function cross-validates [plsRcox](#) models with automatic number of components selection.

### Usage

```

cv.autoplsRcox(
  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 [plsRcox](#)'s `Xplan` argument,
- **time** passed to [plsRcox](#)'s `time` argument,
- **status** [plsRcox](#)'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. If 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 omitted 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 evaluated 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 <a href="#">plsRcox</a> .

### Details

It only computes the recommended *iAUCSH* 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 <i>iAUC_CD</i> for models with 0 to nt components.
cv.error4	Vector with the mean values, across folds, of <i>iAUC_hc</i> for models with 0 to nt components.
cv.error5	Vector with the mean values, across folds, of <i>iAUC_sh</i> for models with 0 to nt components.
cv.error6	Vector with the mean values, across folds, of <i>iAUC_Uno</i> for models with 0 to nt components.
cv.error7	Vector with the mean values, across folds, of <i>iAUC_hz.train</i> for models with 0 to nt components.
cv.error8	Vector with the mean values, across folds, of <i>iAUC_hz.test</i> 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.
folders	Explicit list of the values that were omitted 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 [plsRcox](#)

**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.autoplsRcox.res=cv.autoplsRcox(list(x=X_train_micro,time=Y_train_micro,
status=C_train_micro),nt=3,verbose=FALSE))
```

**Description**

This function cross-validates [coxDKplsDR](#) models.

**Usage**

```

cv.coxDKplsDR(
  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: <ul style="list-style-type: none"> <li>• x the explanatory variables passed to <code>coxDKplsDR</code>'s <code>Xplan</code> argument,</li> <li>• time passed to <code>coxDKplsDR</code>'s <code>time</code> argument,</li> <li>• status <code>coxDKplsDR</code>'s <code>status</code> argument.</li> </ul>
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. If 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 omitted 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 evaluated 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 <code>coxDKplsDR</code> .

**Details**

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

**Value**

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

**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.coxDKplsDR.res=cv.coxDKplsDR(list(x=X_train_micro,time=Y_train_micro,
status=C_train_micro),nt=3))
```

---

cv.coxDKsplSDR

*Cross-validating a DKsplSDR-Model*


---

**Description**

This function cross-validates [coxDKsplSDR](#) models.

**Usage**

```
cv.coxDKsplSDR(
  data,
  method = c("efron", "breslow"),
  nfold = 5,
  nt = 10,
  eta = 0.5,
  plot.it = TRUE,
  se = TRUE,
  givefold,
  scaleX = TRUE,
  scaleY = FALSE,
  folddetails = FALSE,
  allCVcrit = FALSE,
  details = FALSE,
  namedataset = "data",
  save = FALSE,
  verbose = TRUE,
  ...
)
```

**Arguments**

data	A list of three items: <ul style="list-style-type: none"> <li>• x the explanatory variables passed to <code>coxDKsplSDR</code>'s <code>xplan</code> argument,</li> <li>• time passed to <code>coxDKsplSDR</code>'s <code>time</code> argument,</li> <li>• status <code>coxDKsplSDR</code>'s <code>status</code> argument.</li> </ul>
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.
ifold	The number of folds to use to perform the cross-validation process.
nt	The number of components to include in the model. If this is not supplied, 10 components are fitted.
eta	Thresholding parameter. eta should be between 0 and 1.
plot.it	Shall the results be displayed on a plot ?
se	Should standard errors be plotted ?
givefold	Explicit list of omitted values in each fold can be provided using this argument.
scaleX	Shall the predictors be standardized ?
scaleY	Should the time values be standardized ?
folddetails	Should values and completion status for each folds be returned ?
allCVcrit	Should the other 13 CV criteria be evaluated 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 <code>coxDKsplSDR</code> .

**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 <code>iAUC_CD</code> for models with 0 to nt components.
cv.error4	Vector with the mean values, across folds, of <code>iAUC_hc</code> 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.
foldds	Explicit list of the values that were omitted 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 [coxDKsplSDR](#)

### 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) and a grid of eta
(cv.coxDKsplSDR.res=cv.coxDKsplSDR(list(x=X_train_micro,time=Y_train_micro,
status=C_train_micro),nt=3,eta=.1))
```

cv.coxpls

*Cross-validating a Cox-Model fitted on PLSR components***Description**

This function cross-validates `coxpls` models.

**Usage**

```
cv.coxpls(
  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**

<code>data</code>	A list of three items: <ul style="list-style-type: none"> <li>• <code>x</code> the explanatory variables passed to <code>coxpls</code>'s <code>Xplan</code> argument,</li> <li>• <code>time</code> passed to <code>coxpls</code>'s <code>time</code> argument,</li> <li>• <code>status</code> <code>coxpls</code>'s <code>status</code> argument.</li> </ul>
<code>method</code>	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.
<code>nfold</code>	The number of folds to use to perform the cross-validation process.
<code>nt</code>	The number of components to include in the model. If this is not supplied, 10 components are fitted.
<code>plot.it</code>	Shall the results be displayed on a plot ?
<code>se</code>	Should standard errors be plotted ?
<code>givefold</code>	Explicit list of omitted 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 eveled 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 <a href="#">coxpls</a> .

### 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.
folders	Explicit list of the values that were omitted 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 [coxpls](#)

## 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.coxpls.res=cv.coxpls(list(x=X_train_micro,time=Y_train_micro,status=C_train_micro),nt=3))
```

---

cv.coxplsDR

*Cross-validating a plsDR-Model*

---

## Description

This function cross-validates [coxplsDR](#) models.

## Usage

```
cv.coxplsDR(
  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: <ul style="list-style-type: none"> <li>• x the explanatory variables passed to <code>coxplsDR</code>'s <code>Xplan</code> argument,</li> <li>• time passed to <code>coxplsDR</code>'s <code>time</code> argument,</li> <li>• status <code>coxplsDR</code>'s <code>status</code> argument.</li> </ul>
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. If 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 omitted 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 evaluated 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 <code>coxplsDR</code> .

## 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.
folders	Explicit list of the values that were omitted 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 [coxplsDR](#)

**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.coxsplsDR.res=cv.coxsplsDR(list(x=X_train_micro,time=Y_train_micro,status=C_train_micro),nt=3))

```

---

cv.coxsplsDR

*Cross-validating a splsDR-Model*


---

**Description**

This function cross-validates [coxsplsDR](#) models.

**Usage**

```

cv.coxsplsDR(
  data,
  method = c("efron", "breslow"),
  nfold = 5,
  nt = 10,
  eta = 0.5,
  plot.it = TRUE,
  se = TRUE,
  givefold,
  scaleX = TRUE,
  scaleY = FALSE,
  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 [coxsplsDR](#)'s Xplan argument,

- time passed to `coxsplsDR`'s time argument,
- status `coxsplsDR`'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. If this is not supplied, 10 components are fitted.
eta	Thresholding parameter. eta should be between 0 and 1.
plot.it	Shall the results be displayed on a plot ?
se	Should standard errors be plotted ?
givefold	Explicit list of omitted values in each fold can be provided using this argument.
scaleX	Shall the predictors be standardized ?
scaleY	Should the time values be standardized ?
folddetails	Should values and completion status for each folds be returned ?
allCVcrit	Should the other 13 CV criteria be evaluated 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 <code>coxsplsDR</code> .

### 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 <code>iAUC_CD</code> for models with 0 to nt components.
cv.error4	Vector with the mean values, across folds, of <code>iAUC_hc</code> for models with 0 to nt components.
cv.error5	Vector with the mean values, across folds, of <code>iAUC_sh</code> 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.
folders	Explicit list of the values that were omitted 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 [coxsplsDR](#)

### 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) and a grid of eta
(cv.coxsplsDR.res=cv.coxsplsDR(list(x=X_train_micro,time=Y_train_micro,
status=C_train_micro),nt=3,eta=.1))
```

## Description

This function cross-validates [larsDR\\_coxph](#) models.

## Usage

```
cv.larsDR(
  data,
  method = c("efron", "breslow"),
  nfold = 5,
  fraction = seq(0, 1, length = 100),
  plot.it = TRUE,
  se = TRUE,
  givefold,
  scaleX = TRUE,
  scaleY = FALSE,
  folddetails = FALSE,
  allCVcrit = FALSE,
  details = FALSE,
  namedataset = "data",
  save = FALSE,
  verbose = TRUE,
  ...
)
```

## Arguments

data	A list of three items: <ul style="list-style-type: none"> <li>• x the explanatory variables passed to <a href="#">larsDR_coxph</a>'s Xplan argument,</li> <li>• time passed to <a href="#">larsDR_coxph</a>'s time argument,</li> <li>• status <a href="#">larsDR_coxph</a>'s status argument.</li> </ul>
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.
fraction	L1 norm fraction.
plot.it	Shall the results be displayed on a plot ?
se	Should standard errors be plotted ?
givefold	Explicit list of omitted values in each fold can be provided using this argument.

scaleX	Shall the predictors be standardized ?
scaleY	Should the time values be standardized ?
folddetails	Should values and completion status for each folds be returned ?
allCVcrit	Should the other 13 CV criteria be ealed 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 <code>larsDR_coxph</code> .

### Details

It only computes the recommended van Houwelingen CV partial likelihood criterion 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 unsw for models with 0 to nt components.
cv.error12	Vector with the mean values, across folds, of iSchmidScore (robust BS) unsw 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.
fold	Explicit list of the values that were omitted 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.
larsmodfull	Lars model fitted on the residuals.
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.

## See Also

See Also [larsDR\\_coxph](#)

## 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 the default: fraction = seq(0, 1, length = 100)
(cv.larsDR.res=cv.larsDR(list(x=X_train_micro,time=Y_train_micro,
status=C_train_micro),se=TRUE,fraction=seq(0, 1, length = 4)))
```

---

cv.plsRcox

*Cross-validating a plsRcox-Model*

---

## Description

This function cross-validates [plsRcox](#) models.

## Usage

```
cv.plsRcox(
  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: <ul style="list-style-type: none"> <li>• x the explanatory variables passed to <code>plsRcox</code>'s <code>Xplan</code> argument,</li> <li>• time passed to <code>plsRcox</code>'s <code>time</code> argument,</li> <li>• status <code>plsRcox</code>'s <code>status</code> argument.</li> </ul>
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. If 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 omitted 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 evaluated 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 <code>plsRcox</code> .

## Details

It only computes the recommended `iAUCSH` 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 <code>nt</code> 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.
folders	Explicit list of the values that were omitted 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 [plsRcox](#)

### 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.plsRcox.res=cv.plsRcox(list(x=X_train_micro,time=Y_train_micro,status=C_train_micro),nt=3))

```

DKplsRcox

*Partial least squares Regression generalized linear models***Description**

This function implements an extension of Partial least squares Regression to Cox Models.

**Usage**

```

DKplsRcox(Xplan, ...)

DKplsRcoxmodel(Xplan, ...)

## Default S3 method:
DKplsRcoxmodel(
  Xplan,
  time,
  time2,
  event,
  type,
  origin,
  typeres = "deviance",
  collapse,
  weighted,
  scaleX = TRUE,
  scaleY = TRUE,
  nt = min(2, ncol(Xplan)),
  limQ2set = 0.0975,
  dataPredictY = Xplan,
  pvals.expli = FALSE,
  alpha.pvals.expli = 0.05,
  tol_Xi = 10^(-12),
  weights,
  control,
  sparse = FALSE,
  sparseStop = TRUE,
  plot = FALSE,
  allres = FALSE,
  kernel = "rbfdot",

```

```

    hyperkernel,
    verbose = TRUE,
    ...
)

## S3 method for class 'formula'
DKplsRcoxmodel(
  Xplan,
  time,
  time2,
  event,
  type,
  origin,
  typeres = "deviance",
  collapse,
  weighted,
  scaleX = TRUE,
  scaleY = NULL,
  dataXplan = NULL,
  nt = min(2, ncol(Xplan)),
  limQ2set = 0.0975,
  dataPredictY = Xplan,
  pvals.expli = FALSE,
  model_frame = FALSE,
  alpha.pvals.expli = 0.05,
  tol_Xi = 10^(-12),
  weights,
  subset,
  control,
  sparse = FALSE,
  sparseStop = TRUE,
  plot = FALSE,
  allres = FALSE,
  kernel = "rbfdot",
  hyperkernel,
  verbose = TRUE,
  model_matrix = FALSE,
  contrasts.arg = NULL,
  ...
)

```

### Arguments

Xplan	a formula or a matrix with the eXplanatory variables (training) dataset
...	arguments to pass to <code>plsRmodel.default</code> or to <code>plsRmodel.formula</code>
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 ?
nt	number of components to be extracted
limQ2set	limit value for the Q2
dataPredictY	predictor(s) (testing) dataset
pvals.expli	should individual p-values be reported to tune model selection ?
alpha.pvals.expli	level of significance for predictors when pvals.expli=TRUE
tol_Xi	minimal value for Norm2(Xi) and $\det(pp' \times pp)$ if there is any missing value in the dataX. It defaults to $10^{-12}$
weights	an optional vector of 'prior weights' to be used in the fitting process. Should be NULL or a numeric vector.
control	a list of parameters for controlling the fitting process. For <code>glm.fit</code> this is passed to <code>glm.control</code> .
sparse	should the coefficients of non-significant predictors (<alpha.pvals.expli) be set to 0
sparseStop	should component extraction stop when no significant predictors (<alpha.pvals.expli) are found

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.
kernel	<p>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 <a href="#">kernels</a>). The kernLab package provides the most popular kernel functions which can be used by setting the kernel parameter to the following strings:</p> <p><b>list("rbfdot")</b> Radial Basis kernel "Gaussian"  <b>list("polydot")</b> Polynomial kernel  <b>list("vanilladot")</b> Linear kernel  <b>list("tanhdot")</b> Hyperbolic tangent kernel  <b>list("laplacedot")</b> Laplacian kernel  <b>list("besseldot")</b> Bessel kernel  <b>list("anovadot")</b> ANOVA RBF kernel  <b>list("splinedot")</b> Spline kernel</p>
hyperkernel	<p>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 :</p> <ul style="list-style-type: none"> <li>• sigma, inverse kernel width for the Radial Basis kernel function "rbfdot" and the Laplacian kernel "laplacedot".</li> <li>• degree, scale, offset for the Polynomial kernel "polydot".</li> <li>• scale, offset for the Hyperbolic tangent kernel function "tanhdot".</li> <li>• sigma, order, degree for the Bessel kernel "besseldot".</li> <li>• sigma, degree for the ANOVA kernel "anovadot".</li> </ul> <p>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.</p>
verbose	Should some details be displayed ?
dataXplan	an optional data frame, list or environment (or object coercible by <a href="#">as.data.frame</a> 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 coxDKplsDR is called.
model_frame	If TRUE, the model frame is returned.
subset	an optional vector specifying a subset of observations to be used in the fitting process.
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.
method	the method to be used in fitting the model. The default method "glm.fit" uses iteratively reweighted least squares (IWLS). User-supplied fitting functions can be supplied either as a function or a character string naming a function, with a function which takes the same arguments as glm.fit.

**Details**

A typical predictor has the form `response ~ terms` where `response` is the (numeric) response vector and `terms` is a series of terms which specifies a linear predictor for response. A terms specification of the form `first + second` indicates all the terms in `first` together with all the terms in `second` with any duplicates removed.

A specification of the form `first:second` indicates the the set of terms obtained by taking the interactions of all terms in `first` with all terms in `second`. The specification `first*second` indicates the cross of `first` and `second`. This is the same as `first + second + first:second`.

The terms in the formula will be re-ordered so that main effects come first, followed by the interactions, all second-order, all third-order and so on: to avoid this pass a terms object as the formula.

Non-NULL weights can be used to indicate that different observations have different dispersions (with the values in weights being inversely proportional to the dispersions); or equivalently, when the elements of weights are positive integers  $w_i$ , that each response  $y_i$  is the mean of  $w_i$  unit-weight observations.

**Value**

Depends on the model that was used to fit the model.

**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.

**See Also**

[plsR](#) and [plsRglm](#)

**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]
```

```
DKplsRcox(X_train_micro,time=Y_train_micro,event=C_train_micro,nt=5)
DKplsRcox(~X_train_micro,time=Y_train_micro,event=C_train_micro,nt=5)
```

```
DKplsRcox(Xplan=X_train_micro,time=Y_train_micro,event=C_train_micro,nt=5,sparse=TRUE,
alpha.pvals.expli=.15)
DKplsRcox(Xplan=~X_train_micro,time=Y_train_micro,event=C_train_micro,nt=5,sparse=TRUE,
alpha.pvals.expli=.15)
```

---

DR\_coxph

*(Deviance) Residuals Computation*


---

### Description

This function computes the Residuals for a Cox-Model fitted with an intercept as the only explanatory variable. Default behaviour gives the Deviance residuals.

### Usage

```
DR_coxph(
  time,
  time2,
  event,
  type,
  origin,
  typeres = "deviance",
  collapse,
  weighted,
  scaleY = TRUE,
  plot = FALSE,
  ...
)
```

### 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.
typeses	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 ?
...	Arguments to be passed on to survival::coxph.

**Value**

Named num      Vector of the residual values.

**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.

**See Also**

[coxph](#)

**Examples**

```

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

DR_coxph(Y_train_micro,C_train_micro,plot=TRUE)
DR_coxph(Y_train_micro,C_train_micro,scaleY=FALSE,plot=TRUE)
DR_coxph(Y_train_micro,C_train_micro,scaleY=TRUE,plot=TRUE)

rm(Y_train_micro,C_train_micro)

```

---

larsDR\_coxph

*Fitting a LASSO/LARS model on the (Deviance) Residuals*


---

**Description**

This function computes the Cox Model based on lars variables computed model with

- as the response: the Residuals of a Cox-Model fitted with no covariate
- as explanatory variables: Xplan.

It uses the package `lars` to perform PLSR fit.

**Usage**

```

larsDR_coxph(Xplan, ...)

## Default S3 method:
larsDR_coxph(
  Xplan,
  time,
  time2,
  event,
  type,
  origin,
  typeres = "deviance",
  collapse,
  weighted,
  scaleX = FALSE,
  scaleY = TRUE,
  plot = FALSE,
  typelars = "lasso",
  normalize = TRUE,
  max.steps,
  use.Gram = TRUE,
  allres = FALSE,
  verbose = TRUE,

```

```

    ...
)

## S3 method for class 'formula'
larsDR_coxph(
  Xplan,
  time,
  time2,
  event,
  type,
  origin,
  typeres = "deviance",
  collapse,
  weighted,
  scaleX = FALSE,
  scaleY = TRUE,
  plot = FALSE,
  typelars = "lasso",
  normalize = TRUE,
  max.steps,
  use.Gram = TRUE,
  allres = FALSE,
  dataXplan = NULL,
  subset,
  weights,
  model_frame = FALSE,
  model_matrix = FALSE,
  verbose = TRUE,
  contrasts.arg = NULL,
  ...
)

```

### Arguments

Xplan	a formula or a matrix with the eXplanatory variables (training) dataset
...	Arguments to be passed on to <code>survival::coxph</code> or to <code>lars::lars</code> .
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.
typeses	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 ?
plot	Should the survival function be plotted ?
typelars	One of "lasso", "lar", "forward.stagewise" or "stepwise". The names can be abbreviated to any unique substring. Default is "lasso".
normalize	If TRUE, each variable is standardized to have unit L2 norm, otherwise it is left alone. Default is TRUE.
max.steps	Limit the number of steps taken; the default is $8 * \min(m, n - \text{intercept})$ , with m the number of variables, and n the number of samples. For type="lar" or type="stepwise", the maximum number of steps is $\min(m, n - \text{intercept})$ . For type="lasso" and especially type="forward.stagewise", there can be many more terms, because although no more than $\min(m, n - \text{intercept})$ variables can be active during any step, variables are frequently dropped and added as the algorithm proceeds. Although the default usually guarantees that the algorithm has proceeded to the saturated fit, users should check.
use.Gram	When the number m of variables is very large, i.e. larger than N, then you may not want LARS to precompute the Gram matrix. Default is use.Gram=TRUE
allres	FALSE to return only the Cox model and TRUE for additional results. See details. Defaults to FALSE.
verbose	Should some details be displayed ?
dataXplan	an optional data frame, list or environment (or object coercible by <code>as.data.frame</code> 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 plscox is called.
subset	an optional vector specifying a subset of observations to be used in the fitting process.

<code>weights</code>	an optional vector of 'prior weights' to be used in the fitting process. Should be NULL or a numeric vector.
<code>model_frame</code>	If TRUE, the model frame is returned.
<code>model_matrix</code>	If TRUE, the model matrix is returned.
<code>contrasts.arg</code>	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

This function computes the LASSO/LARS model with the Residuals of a Cox-Model fitted with an intercept as the only explanatory variable as the response and Xplan as explanatory variables. Default behaviour uses the Deviance residuals.

If `allres=FALSE` returns only the final Cox-model. If `allres=TRUE` returns a list with the (Deviance) Residuals, the LASSO/LARS model fitted to the (Deviance) Residuals, the eXplanatory variables and the final Cox-model. `allres=TRUE` is useful for evaluating model prediction accuracy on a test sample.

### Value

If `allres=FALSE` :

`cox_larsDR` Final Cox-model.

If `allres=TRUE` :

`DR_coxph` The (Deviance) Residuals.

`larsDR` The LASSO/LARS model fitted to the (Deviance) Residuals.

`X_larsDR` The eXplanatory variables.

`cox_larsDR` Final Cox-model.

### 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.

### See Also

[coxph](#), [lars](#)

## 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_larsDR_fit <- larsDR_coxph(X_train_micro,Y_train_micro,C_train_micro,max.steps=6,
use.Gram=FALSE,scaleX=TRUE))
(cox_larsDR_fit <- larsDR_coxph(~X_train_micro,Y_train_micro,C_train_micro,max.steps=6,
use.Gram=FALSE,scaleX=TRUE))
(cox_larsDR_fit <- larsDR_coxph(~.,Y_train_micro,C_train_micro,max.steps=6,
use.Gram=FALSE,scaleX=TRUE,dataXplan=X_train_micro_df))

larsDR_coxph(~X_train_micro,Y_train_micro,C_train_micro,max.steps=6,use.Gram=FALSE)
larsDR_coxph(~X_train_micro,Y_train_micro,C_train_micro,max.steps=6,use.Gram=FALSE,scaleX=FALSE)
larsDR_coxph(~X_train_micro,Y_train_micro,C_train_micro,max.steps=6,use.Gram=FALSE,
scaleX=TRUE,allres=TRUE)

rm(X_train_micro,Y_train_micro,C_train_micro,cox_larsDR_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 Guenet, *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)
```

---

plsRcox

*Partial least squares Regression generalized linear models*

---

## Description

This function implements an extension of Partial least squares Regression to Cox Models.

## Usage

```
plsRcox(Xplan, ...)

plsRcoxmodel(Xplan, ...)

## Default S3 method:
plsRcoxmodel(
  Xplan,
  time,
  time2,
  event,
  type,
```

```
origin,  
typeres = "deviance",  
collapse,  
weighted,  
scaleX = TRUE,  
scaleY = TRUE,  
nt = min(2, ncol(Xplan)),  
limQ2set = 0.0975,  
dataPredictY = Xplan,  
pvals.expli = FALSE,  
alpha.pvals.expli = 0.05,  
tol_Xi = 10^(-12),  
weights,  
control,  
sparse = FALSE,  
sparseStop = TRUE,  
allres = TRUE,  
verbose = TRUE,  
...  
)  
  
## S3 method for class 'formula'  
plsRcoxmodel(  
  Xplan,  
  time,  
  time2,  
  event,  
  type,  
  origin,  
  typeres = "deviance",  
  collapse,  
  weighted,  
  scaleX = TRUE,  
  scaleY = NULL,  
  dataXplan = NULL,  
  nt = min(2, ncol(Xplan)),  
  limQ2set = 0.0975,  
  dataPredictY = Xplan,  
  pvals.expli = FALSE,  
  model_frame = FALSE,  
  alpha.pvals.expli = 0.05,  
  tol_Xi = 10^(-12),  
  weights,  
  subset,  
  control,  
  sparse = FALSE,  
  sparseStop = TRUE,  
  allres = TRUE,
```

```

    verbose = TRUE,
    model_matrix = FALSE,
    contrasts.arg = NULL,
    ...
)

```

## Arguments

<code>Xplan</code>	a formula or a matrix with the eXplanatory variables (training) dataset
<code>...</code>	arguments to pass to <code>plsRmodel.default</code> or to <code>plsRmodel.formula</code>
<code>time</code>	for right censored data, this is the follow up time. For interval data, the first argument is the starting time for the interval.
<code>time2</code>	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.
<code>event</code>	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, ( <code>start</code> , <code>end</code> ]. For counting process data, event indicates whether an event occurred at the end of the interval.
<code>type</code>	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 <code>time2</code> argument is absent or present, respectively.
<code>origin</code>	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.
<code>typeres</code>	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.
<code>collapse</code>	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 <code>collapse=c(1, 1, 1, 2, 3, 3, 4, 4, 4, 4)</code> could be used to obtain per subject rather than per observation residuals.
<code>weighted</code>	if TRUE and the model was fit with case weights, then the weighted residuals are returned.
<code>scaleX</code>	Should the <code>Xplan</code> columns be standardized ?
<code>scaleY</code>	Should the <code>time</code> values be standardized ?
<code>nt</code>	number of components to be extracted
<code>limQ2set</code>	limit value for the Q2
<code>dataPredictY</code>	predictor(s) (testing) dataset

<code>pvals.expli</code>	should individual p-values be reported to tune model selection ?
<code>alpha.pvals.expli</code>	level of significance for predictors when <code>pvals.expli=TRUE</code>
<code>tol_Xi</code>	minimal value for $\text{Norm2}(X_i)$ and $\det(pp' \times pp)$ if there is any missing value in the data $X$ . It defaults to $10^{-12}$
<code>weights</code>	an optional vector of 'prior weights' to be used in the fitting process. Should be NULL or a numeric vector.
<code>control</code>	a list of parameters for controlling the fitting process. For <code>glm.fit</code> this is passed to <code>glm.control</code> .
<code>sparse</code>	should the coefficients of non-significant predictors ( <code>&lt;alpha.pvals.expli</code> ) be set to 0
<code>sparseStop</code>	should component extraction stop when no significant predictors ( <code>&lt;alpha.pvals.expli</code> ) are found
<code>allres</code>	FALSE to return only the Cox model and TRUE for additionnal results. See details. Defaults to FALSE.
<code>verbose</code>	Should some details be displayed ?
<code>dataXplan</code>	an optional data frame, list or environment (or object coercible by <code>as.data.frame</code> to a data frame) containing the variables in the model. If not found in <code>dataXplan</code> , the variables are taken from <code>environment(Xplan)</code> , typically the environment from which <code>coxDKplsDR</code> is called.
<code>model_frame</code>	If TRUE, the model frame is returned.
<code>subset</code>	an optional vector specifying a subset of observations to be used in the fitting process.
<code>model_matrix</code>	If TRUE, the model matrix is returned.
<code>contrasts.arg</code>	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.
<code>method</code>	the method to be used in fitting the model. The default method " <code>glm.fit</code> " uses iteratively reweighted least squares (IWLS). User-supplied fitting functions can be supplied either as a function or a character string naming a function, with a function which takes the same arguments as <code>glm.fit</code> .

## Details

A typical predictor has the form `response ~ terms` where `response` is the (numeric) response vector and `terms` is a series of terms which specifies a linear predictor for response. A terms specification of the form `first + second` indicates all the terms in `first` together with all the terms in `second` with any duplicates removed.

A specification of the form `first:second` indicates the the set of terms obtained by taking the interactions of all terms in `first` with all terms in `second`. The specification `first*second` indicates the cross of `first` and `second`. This is the same as `first + second + first:second`.

The terms in the formula will be re-ordered so that main effects come first, followed by the interactions, all second-order, all third-order and so on: to avoid this pass a terms object as the formula.

Non-NULL weights can be used to indicate that different observations have different dispersions (with the values in weights being inversely proportional to the dispersions); or equivalently, when the elements of weights are positive integers  $w_i$ , that each response  $y_i$  is the mean of  $w_i$  unit-weight observations.

### Value

Depends on the model that was used to fit the model.

### 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.

### See Also

[plsR](#) and [plsRglm](#)

### 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]

plsRcox(X_train_micro,time=Y_train_micro,event=C_train_micro,nt=5)
plsRcox(~X_train_micro,time=Y_train_micro,event=C_train_micro,nt=5)

plsRcox(Xplan=X_train_micro,time=Y_train_micro,event=C_train_micro,nt=5,sparse=TRUE,
alpha.pvals.expli=.15)
plsRcox(Xplan=~X_train_micro,time=Y_train_micro,event=C_train_micro,nt=5,sparse=TRUE,
alpha.pvals.expli=.15)
```

---

predict.coxDRmodel      *Predict method for DR-based Cox component models*

---

### Description

This function provides prediction methods for the rich objects returned by `coxplsDR(..., allres = TRUE)`, `coxspplsDR(..., allres = TRUE)` and `coxDKspplsDR(..., allres = TRUE)`.

### Usage

```
## S3 method for class 'coxplsDRmodel'
predict(
  object,
  newdata,
  comps = ncol(object$tt_plsDR),
  type = c("lp", "risk", "expected", "terms", "scores"),
  se.fit = FALSE,
  reference = c("strata", "sample", "zero"),
  y = NULL,
  weights = NULL,
  verbose = TRUE,
  ...
)

## S3 method for class 'coxspplsDRmodel'
predict(
  object,
  newdata,
  comps = ncol(object$tt_spplsDR),
  type = c("lp", "risk", "expected", "terms", "scores"),
  se.fit = FALSE,
  reference = c("strata", "sample", "zero"),
  y = NULL,
  weights = NULL,
  verbose = TRUE,
  ...
)

## S3 method for class 'coxDKspplsDRmodel'
predict(
  object,
  newdata,
  comps = ncol(object$tt_DKspplsDR),
  type = c("lp", "risk", "expected", "terms", "scores"),
  se.fit = FALSE,
  reference = c("strata", "sample", "zero"),
  y = NULL,
```

```

weights = NULL,
verbose = TRUE,
...
)

```

### Arguments

<code>object</code>	An object returned by one of the DR-based fitting functions with <code>allres = TRUE</code> .
<code>newdata</code>	An optional data frame or matrix containing original covariates. If omitted, predictions are computed on the training data.
<code>comps</code>	Number of latent components to use for prediction.
<code>type</code>	Type of predicted value. Choices are the linear predictor ("lp"), the risk score <code>exp(lp)</code> ("risk"), the expected number of events ("expected"), the terms of the linear predictor ("terms") or the latent component scores ("scores").
<code>se.fit</code>	If TRUE, pointwise standard errors are produced by the underlying Cox model.
<code>reference</code>	Reference level used to center relative predictions. This is passed to <a href="#">predict.coxph</a> and affects <code>type = "lp"</code> , "risk" and "terms".
<code>y</code>	Optional <a href="#">Surv</a> response to use with <code>type = "expected"</code> when predicting on genuinely new observations.
<code>weights</code>	Optional case weights used when rebuilding a model matrix for formula-based fits.
<code>verbose</code>	Should some details be displayed?
<code>...</code>	Additional arguments passed to <a href="#">predict.coxph</a> .

### Value

A vector, matrix or list of predictions depending on `type` and `se.fit`.

### See Also

[predict.coxph](#), [coxplsDR](#), [coxspplsDR](#), [coxDKspplsDR](#)

---

`predict.plsRcoxmodel` *Print method for plsRcox models*

---

### Description

This function provides a `predict` method for the class "plsRcoxmodel"

**Usage**

```
## S3 method for class 'plsRcoxmodel'
predict(
  object,
  newdata,
  comps = object$computed_nt,
  type = c("lp", "risk", "expected", "terms", "scores"),
  se.fit = FALSE,
  reference = c("strata", "sample", "zero"),
  y = NULL,
  weights,
  methodNA = "adaptative",
  verbose = TRUE,
  ...
)
```

**Arguments**

object	An object of the class "plsRcoxmodel".
newdata	An optional data frame in which to look for variables with which to predict. If omitted, the fitted values are used. For type = "expected" on genuinely new observations, supply the associated survival response with y; when newdata is a subset of the training rows and keeps compatible row names, the training response is matched automatically.
comps	A value with a single value of component to use for prediction.
type	Type of predicted value. Choices are the linear predictor ("lp"), the risk score exp(lp) ("risk"), the expected number of events given the covariates and follow-up time ("expected"), the terms of the linear predictor ("terms") or the scores ("scores").
se.fit	If TRUE, pointwise standard errors are produced for the predictions using the Cox model.
reference	Reference level used to center relative predictions. This is passed to <a href="#">predict.coxph</a> and affects type = "lp", "risk" and "terms".
y	Optional <a href="#">Surv</a> response to use with type = "expected" when predicting on new observations. The time values are interpreted on the original scale used to fit the model and are rescaled internally when needed.
weights	Vector of case weights. If weights is a vector of integers, then the estimated coefficients are equivalent to estimating the model from data with the individual cases replicated as many times as indicated by weights.
methodNA	Selects the way of predicting the response or the scores of the new data. For complete rows, without any missing value, there are two different ways of computing the prediction. As a consequence, for mixed datasets, with complete and incomplete rows, there are two ways of computing prediction : either predicts any row as if there were missing values in it ( <code>missingdata</code> ) or selects the prediction method accordingly to the completeness of the row ( <code>adaptative</code> ).
verbose	Should some details be displayed ?

... Additional arguments passed on to `predict.coxph`.

### Value

When type is "response", a matrix of predicted response values is returned.  
When type is "scores", a score matrix is returned.

### 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.

### See Also

`predict.coxph`

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

modpls <- plsRcox(X_train_micro,time=Y_train_micro,event=C_train_micro,nt=3)

predict(modpls)
#Identical to predict(modpls,type="lp")

predict(modpls,type="risk")
predict(modpls,type="expected")
predict(modpls,type="terms")
predict(modpls,type="scores")

predict(modpls,se.fit=TRUE)
#Identical to predict(modpls,type="lp")
predict(modpls,type="risk",se.fit=TRUE)
predict(modpls,type="expected",se.fit=TRUE)
predict(modpls,type="terms",se.fit=TRUE)
predict(modpls,type="scores",se.fit=TRUE)
```

```

#Identical to predict(modpls,type="lp")
predict(modpls,newdata=X_train_micro[1:5,],type="risk")
predict(modpls,newdata=X_train_micro[1:5,],type="expected")
predict(modpls,newdata=X_train_micro[1:5,],type="terms")
predict(modpls,newdata=X_train_micro[1:5,],type="scores")

#Identical to predict(modpls,type="lp")
predict(modpls,newdata=X_train_micro[1:5,],type="risk",se.fit=TRUE)
predict(modpls,newdata=X_train_micro[1:5,],type="expected",se.fit=TRUE)
predict(modpls,newdata=X_train_micro[1:5,],type="terms",se.fit=TRUE)
predict(modpls,newdata=X_train_micro[1:5,],type="scores")

newY_micro <- survival::Surv(Y_train_micro[1:5], C_train_micro[1:5])
predict(modpls,newdata=unname(X_train_micro[1:5,]),type="expected",y=newY_micro)

predict(modpls,newdata=X_train_micro[1:5,],type="risk",comps=1)
predict(modpls,newdata=X_train_micro[1:5,],type="risk",comps=2)
predict(modpls,newdata=X_train_micro[1:5,],type="risk",comps=3)
try(predict(modpls,newdata=X_train_micro[1:5,],type="risk",comps=4))

predict(modpls,newdata=X_train_micro[1:5,],type="terms",comps=1)
predict(modpls,newdata=X_train_micro[1:5,],type="terms",comps=2)
predict(modpls,newdata=X_train_micro[1:5,],type="terms",comps=3)
try(predict(modpls,newdata=X_train_micro[1:5,],type="terms",comps=4))

predict(modpls,newdata=X_train_micro[1:5,],type="scores",comps=1)
predict(modpls,newdata=X_train_micro[1:5,],type="scores",comps=2)
predict(modpls,newdata=X_train_micro[1:5,],type="scores",comps=3)
try(predict(modpls,newdata=X_train_micro[1:5,],type="scores",comps=4))

```

---

print.plsRcoxmodel      *Print method for plsRcox models*

---

## Description

This function provides a print method for the class "plsRcoxmodel"

## Usage

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

## Arguments

x	an object of the class "plsRcoxmodel"
...	not used

**Value**

NULL

**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.

**See Also**[print](#)**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,]
Y_train_micro <- micro.censure$survyear[1:80]
C_train_micro <- micro.censure$DC[1:80]

modpls <- plsRcox(X_train_micro,time=Y_train_micro,event=C_train_micro,nt=3)
print(modpls)
```

---

```
print.summary.plsRcoxmodel
```

*Print method for summaries of plsRcox models*

---

**Description**

This function provides a print method for the class "summary.plsRcoxmodel"

**Usage**

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

**Arguments**

x                    an object of the class "summary.plsRcoxmodel"  
...                   not used

**Value**

language            call of the model

**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.

**See Also**

[print](#) and [summary](#)

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

modpls <- plsRcox(X_train_micro,time=Y_train_micro,event=C_train_micro,nt=3)
print(summary(modpls))
```

---

summary.plsRcoxmodel    *Summary method for plsRcox models*

---

**Description**

This function provides a summary method for the class "plsRcoxmodel"

**Usage**

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

**Arguments**

object            an object of the class "plsRcoxmodel"  
...               further arguments to be passed to or from methods.

**Value**

call             function call of plsRcox models

**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.

**See Also**

[summary](#)

**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,]  
Y_train_micro <- micro.censure$survyear[1:80]  
C_train_micro <- micro.censure$DC[1:80]  
  
modpls <- plsRcox(X_train_micro,time=Y_train_micro,event=C_train_micro,nt=3)  
summary(modpls)
```

---

`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 Guenet, *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**
  - micro.censure, 92
  - Xmicro.censure\_compl\_imp, 107
- \* **methods**
  - predict.plsRcoxmodel, 100
  - print.plsRcoxmodel, 103
  - print.summary.plsRcoxmodel, 104
  - summary.plsRcoxmodel, 105
- \* **models**
  - coxDKpls2DR, 2
  - coxDKplsDR, 7
  - coxDKsplDR, 12
  - coxpls, 17
  - coxpls2, 21
  - coxpls2DR, 24
  - coxpls3, 28
  - coxpls3DR, 32
  - coxplsDR, 36
  - coxspplsDR, 40
  - cv.autopplsRcox, 44
  - cv.coxDKplsDR, 48
  - cv.coxDKsplDR, 53
  - cv.coxpls, 58
  - cv.coxplsDR, 62
  - cv.coxspplsDR, 67
  - cv.larsDR, 72
  - cv.plsRcox, 76
  - DKplsRcox, 81
  - DR\_coxph, 86
  - larsDR\_coxph, 88
  - plsRcox, 94
  - coxDKplsDR, 7
  - coxDKsplDR, 12
  - coxpls, 17
  - coxpls2, 21
  - coxpls2DR, 24
  - coxpls3, 28
  - coxpls3DR, 32
  - coxplsDR, 36
  - coxspplsDR, 40
  - cv.autopplsRcox, 44
  - cv.coxDKplsDR, 48
  - cv.coxDKsplDR, 53
  - cv.coxpls, 58
  - cv.coxplsDR, 62
  - cv.coxspplsDR, 67
  - cv.larsDR, 72
  - cv.plsRcox, 76
  - DKplsRcox, 81
  - DR\_coxph, 86
  - larsDR\_coxph, 88
  - plsRcox, 94
- as.data.frame, 5, 10, 15, 19, 23, 27, 31, 35, 39, 42, 84, 90, 97
- coxDKpls2DR, 2
- coxDKplsDR, 7, 48, 49, 53
- coxDKsplDR, 12, 53, 54, 57, 100
- coxph, 6, 11, 16, 20, 24, 28, 32, 36, 40, 43, 87, 91
- coxpls, 17, 58, 59, 62
- coxpls2, 21
- coxpls2DR, 24
- coxpls3, 28
- coxpls3DR, 32
- coxplsDR, 36, 62, 63, 66, 100
- coxspplsDR, 40, 67, 68, 71, 100
- cv.autopplsRcox, 44
- cv.coxDKplsDR, 48
- cv.coxDKsplDR, 53

cv.coxpls, 58  
cv.coxplsDR, 62  
cv.coxsplsDR, 67  
cv.larsDR, 72  
cv.plsRcox, 76

DKplsRcox, 81  
DKplsRcoxmodel (DKplsRcox), 81  
DR\_coxph, 86

glm.control, 83, 97

kernels, 5, 10, 15, 84

lars, 91  
larsDR\_coxph, 72, 73, 76, 88

micro.censure, 92  
mvrCv, 5, 23, 26, 27

pls, 10, 15, 19, 38, 42  
PLS\_lm, 32, 36  
plsR, 85, 98  
plsr, 6, 11, 16, 20, 24, 28, 40, 43  
plsRcox, 44, 45, 48, 76, 77, 80, 94  
plsRcoxmodel (plsRcox), 94  
plsRglm, 85, 98  
predict.coxDKsplsDRmodel  
    (predict.coxDRmodel), 99  
predict.coxDRmodel, 99  
predict.coxph, 100–102  
predict.coxplsDRmodel  
    (predict.coxDRmodel), 99  
predict.coxsplsDRmodel  
    (predict.coxDRmodel), 99  
predict.plsRcoxmodel, 100  
print, 104, 105  
print.plsRcoxmodel, 103  
print.summary.plsRcoxmodel, 104

summary, 105, 106  
summary.plsRcoxmodel, 105  
Surv, 100, 101

Xmicro.censure\_compl\_imp, 107