Package 'Coxmos'

March 25, 2024

Title Cox MultiBlock Survival

Version 1.0.2

Maintainer Pedro Salguero García <pedrosalguerog@gmail.com>

Description This software package provides Cox survival analysis for high-dimensional and multiblock datasets.
It encompasses a suite of functions dedicated from the classical Cox regression to newest analysis, including Cox proportional hazards model, Stepwise Cox regression, and Elastic-Net Cox regression,
Sparse Partial Least Squares Cox regression (sPLS-COX) incorporating three distinct strategies, and two Multiblock-PLS Cox regression (MB-sPLS-COX) methods. This tool is designed to adeptly handle high-dimensional data, and provides tools for cross-validation, plot generation, and additional resources for interpreting results. While references are available within the corresponding functions, key literature is mentioned below.
Terry M Therneau (2024) <https://CRAN.R-project.org/package=survival>,
Noah Simon et al. (2011) <doi:10.18637/jss.v039.i05>,
Philippe Bastien et al. (2005) <doi:10.1016/j.csda.2004.02.005>,
Philippe Bastien (2008) <doi:10.1016/j.chemolab.2007.09.009>,
Philippe Bastien et al. (2014) <doi:10.1093/bioinformatics/btu660>,
Kassu Mehari Beyene and Anouar El Ghouch (2020) <doi:10.1002/sim.8671>,
Florian Rohart et al. (2017) <doi:10.1371/journal.pcbi.1005752>.

URL https://github.com/BiostatOmics/Coxmos

BugReports https://github.com/BiostatOmics/Coxmos/issues

License CC BY 4.0

Encoding UTF-8

RoxygenNote 7.2.3

biocViews

Depends R (>= 4.1.0),

Imports caret, cowplot, furrr, future, ggrepel, ggplot2, ggpubr, glmnet, MASS, mixOmics, progress, purrr, Rdpack, scattermore, stats, survcomp, survival, survminer, svglite, tidyr, utils

Suggests nsROC, smoothROCtime, survivalROC, risksetROC, ggforce, knitr, RColorConesa, rmarkdown

RdMacros Rdpack

LazyData true

NeedsCompilation yes

VignetteBuilder knitr

Author Pedro Salguero García [aut, cre, rev] (<https://orcid.org/0000-0002-1879-3374>), Sonia Tarazona Campos [ths], Ana Conesa Cegarra [ths], Kassu Mehari Beyene [ctb], Luis Meira Machado [ctb], Marta Sestelo [ctb], Artur Araújo [ctb]

Repository CRAN

Date/Publication 2024-03-25 20:32:38 UTC

R topics documented:

`Beran` *Estimation of the conditional distribution function of the response, given the covariate under random censoring.*

Description

Computes the conditional survival probability P(T > y|Z = z)

Usage

```
Beran(
  time,
  status,
  covariate,
  delta,
  x,
  y,
  kernel = "gaussian",
  bw,
  lower.tail = FALSE
)
```

Arguments

`time`	The survival time of the process.
`status`	Censoring indicator of the total time of the process; 0 if the total time is censored and 1 otherwise.
`covariate`	Covariate values for obtaining estimates for the conditional probabilities.
`delta`	Censoring indicator of the covariate.
`x`	The first time (or covariate value) for obtaining estimates for the conditional probabilities. If missing, 0 will be used.
`y`	The total time for obtaining estimates for the conditional probabilities.
`kernel`	A character string specifying the desired kernel. See details below for possible options. Defaults to "gaussian" where the gaussian density kernel will be used.
`bw`	A single numeric value to compute a kernel density bandwidth.
`lower.tail`	logical; if FALSE (default), probabilities are P(T > y|Z = z) otherwise, P(T <= y|Z = z).

Details

Possible options for argument window are "gaussian", "epanechnikov", "tricube", "boxcar", "triangular", "quartic" or "cosine".

Author(s)

Luis Meira-Machado and Marta Sestelo

References

R. Beran. Nonparametric regression with randomly censored survival data. Technical report, University of California, Berkeley, 1981.

`cenROC` *Estimation of the time-dependent ROC curve for right censored survival data*

Description

This function computes the time-dependent ROC curve for right censored survival data using the cumulative sensitivity and dynamic specificity definitions. The ROC curves can be either empirical (non-smoothed) or smoothed with/wtihout boundary correction. It also calculates the time-dependent area under the ROC curve (AUC). Edited by Pedro Salguero to remove the PLOT argument.

Usage

```
cenROC(Y, M, censor, t, U = NULL, h = NULL, bw = "NR", method = "tra",
    ktype = "normal", ktype1 = "normal", B = 0, alpha = 0.05, plot = FALSE)
```

Arguments

`Y`	The numeric vector of event-times or observed times.
`M`	The numeric vector of marker values for which the time-dependent ROC curves is computed.
`censor`	The censoring indicator, `1` if event, `0` otherwise.
`t`	A scaler time point at which the time-dependent ROC curve is computed.
`U`	The vector of grid points where the ROC curve is estimated. The default is a sequence of `151` numbers between `0` and `1`.
`h`	A scaler for the bandwidth of Beran's weight calculaions. The default is the value obtained by using the method of Sheather and Jones (1991).
`bw`	A character string specifying the bandwidth estimation method for the ROC itself. The possible options are `"NR"` for the normal reference, the plug-in `"PI"` and the cross-validation `"CV"`. The default is the `"NR"` normal reference method. The user can also introduce a numerical value.

method	The method of ROC curve estimation. The possible options are "emp" emperical method; "untra" smooth without boundary correction and "tra" is smooth ROC curve estimation with boundary correction. The default is the "tra" smooth ROC curve estimate with boundary correction.
ktype	A character string giving the type kernel distribution to be used for smoothing the ROC curve: "normal", "epanechnikov", "biweight", or "triweight". By default, the "normal" kernel is used.
ktype1	A character string specifying the desired kernel needed for Beran weight calculation. The possible options are "normal", "epanechnikov", "tricube", "boxcar", "triangular", or "quartic". The defaults is "normal" kernel density.
B	The number of bootstrap samples to be used for variance estimation. The default is 0, no variance estimation.
alpha	The significance level. The default is 0.05.
plot	The logical parameter to see the ROC curve plot. The default is TRUE. Currently disabled.

Details

The empirical (non-smoothed) ROC estimate and the smoothed ROC estimate with/without boundary correction can be obtained using this function. The smoothed ROC curve estimators require selecting two bandwidth parametrs: one for Beran's weight calculation and one for smoothing the ROC curve. For the latter, three data-driven methods: the normal reference "NR", the plug-in "PI" and the cross-validation "CV" were implemented. To select the bandwidth parameter needed for Beran's weight calculation, by default, the plug-in method of Sheather and Jones (1991) is used but it is also possible introduce a numeric value. See Beyene and El Ghouch (2020) for details.

Value

Returns the following items:

ROC The vector of estimated ROC values. These will be numeric numbers between zero and one.

U The vector of grid points used.

AUC A data frame of dimension 1×4. The columns are: AUC, standard error of AUC, the lower and upper limits of bootstrap CI.

bw The computed value of bandwidth. For the empirical method this is always NA.

Dt The vector of estimated event status.

M The vector of Marker values.

Author(s)

Kassu Mehari Beyene, Catholic University of Louvain. <kasu.beyene@uclouvain.be>

Anouar El Ghouch, Catholic University of Louvain. <anouar.elghouch@uclouvain.be>

References

Beyene, K. M. and El Ghouch A. (2020). Smoothed time-dependent ROC curves for right-censored survival data. *submitted*.

Sheather, S. J. and Jones, M. C. (1991). A Reliable data-based bandwidth selection method for kernel density estimation. *Journal of the Royal Statistical Society*. Series B (Methodological) 53(3): 683–690.

`cox` *cox*

Description

The `cox` function conducts a Cox proportional hazards regression analysis, a type of survival analysis. It is designed to handle right-censored data and is built upon the `coxph` function from the `survival` package. The function returns an object of class "Coxmos" with the attribute model labeled as "cox".

Usage

```
cox(
  X,
  Y,
  x.center = TRUE,
  x.scale = FALSE,
  remove_near_zero_variance = TRUE,
  remove_zero_variance = FALSE,
  toKeep.zv = NULL,
  remove_non_significant = FALSE,
  alpha = 0.05,
  MIN_EPV = 5,
  FORCE = FALSE,
  returnData = TRUE,
  verbose = FALSE
)
```

Arguments

`X`	Numeric matrix or data.frame. Explanatory variables. Qualitative variables must be transform into binary variables.
`Y`	Numeric matrix or data.frame. Response variables. Object must have two columns named as "time" and "event". For event column, accepted values are: 0/1 or FALSE/TRUE for censored and event observations.
`x.center`	Logical. If x.center = TRUE, X matrix is centered to zero means (default: TRUE).
`x.scale`	Logical. If x.scale = TRUE, X matrix is scaled to unit variances (default: FALSE).

`remove_near_zero_variance`
: Logical. If remove_near_zero_variance = TRUE, near zero variance variables will be removed (default: TRUE).

`remove_zero_variance`
: Logical. If remove_zero_variance = TRUE, zero variance variables will be removed (default: TRUE).

`toKeep.zv`
: Character vector. Name of variables in X to not be deleted by (near) zero variance filtering (default: NULL).

`remove_non_significant`
: Logical. If remove_non_significant = TRUE, non-significant variables/components in final cox model will be removed until all variables are significant by forward selection (default: FALSE).

`alpha`
: Numeric. Numerical values are regarded as significant if they fall below the threshold (default: 0.05).

`MIN_EPV`
: Numeric. Minimum number of Events Per Variable (EPV) you want reach for the final cox model. Used to restrict the number of variables/components can be computed in final cox models. If the minimum is not meet, the model cannot be computed (default: 5).

`FORCE`
: Logical. In case the MIN_EPV is not meet, it allows to compute the model (default: FALSE).

`returnData`
: Logical. Return original and normalized X and Y matrices (default: TRUE).

`verbose`
: Logical. If verbose = TRUE, extra messages could be displayed (default: FALSE).

Details

The Cox proportional hazards regression model is a linear model that describes the relationship between the hazard rate and one or more predictor variables. The function provided here offers several preprocessing steps to ensure the quality and robustness of the model.

The function allows for the centering and scaling of predictor variables, which can be essential for the stability and interpretability of the model. It also provides options to remove variables with near-zero or zero variance, which can be problematic in regression analyses. Such variables offer little to no information and can lead to overfitting.

Another notable feature is the ability to remove non-significant predictors from the final model through a backward selection process. This ensures that only variables that contribute significantly to the model are retained.

The function also checks for the minimum number of events per variable (EPV) to ensure the robustness of the model. If the specified EPV is not met, the function can either halt the computation or proceed based on user preference.

It's important to note that while this function is tailored for standard Cox regression, it might not be suitable for high-dimensional data. In such cases, users are advised to consider alternative methods like `coxEN()` or PLS-based Cox methods.

Value

Instance of class "Coxmos" and model "cox". The class contains the following elements:

`X`: List of normalized X data information.

- (data): normalized X matrix
- (x.mean): mean values for X matrix
- (x.sd): standard deviation for X matrix

Y: List of normalized Y data information.

- (data): normalized Y matrix
- (y.mean): mean values for Y matrix
- (y.sd): standard deviation for Y matrix

survival_model: List of survival model information

- fit: coxph object.
- AIC: AIC of cox model.
- BIC: BIC of cox model.
- lp: linear predictors for train data.
- coef: Coefficients for cox model.
- YChapeau: Y Chapeau residuals.
- Yresidus: Y residuals.

call: call function

X_input: X input matrix

Y_input: Y input matrix

nsv: Variables removed by remove_non_significant if any.

nzv: Variables removed by remove_near_zero_variance or remove_zero_variance.

nz_coeffvar: Variables removed by coefficient variation near zero.

removed_variables_correlation: Variables removed by being high correlated with other variables.

class: Model class.

time: time consumed for running the cox analysis.

Author(s)

Pedro Salguero Garcia. Maintainer: pedsalga@upv.edu.es

References

Cox D (1972). "Regression models and life tables (with discussion." *Royal Statistical Society*. https://doi.org/10.1111/j.2517-6161.1972.tb00899.x. Concato J, Peduzzi P, Holford TR, Feinstein AR (1995). "Importance of events per independent variable in proportional hazards analysis I. Background, goals, and general strategy." *Journal of Clinical Epidemiology*. doi:10.1016/08954356(95)005102, https://pubmed.ncbi.nlm.nih.gov/8543963/. Therneau TM (2024). *A Package for Survival Analysis in R*. R package version 3.5-8, https://CRAN.R-project.org/package=survival.

Examples

```
data("X_proteomic")
data("Y_proteomic")
X <- X_proteomic[,1:10]
Y <- Y_proteomic
cox(X, Y, x.center = TRUE, x.scale = TRUE)
```

`cox.prediction` *cox.prediction*

Description

The `cox.prediction` function facilitates Cox predictions based on a given Coxmos model, specifically tailored for raw data input. It seamlessly integrates the generation of a score matrix, especially when a PLS Survival analysis has been executed, and subsequently conducts the Cox prediction. The function offers flexibility in prediction types and methods, catering to diverse analytical requirements.

Usage

```
cox.prediction(model, new_data, time = NULL, type = "lp", method = "cox")
```

Arguments

`model`	Coxmos model.
`new_data`	Numeric matrix or data.frame. New explanatory variables (raw data). Qualitative variables must be transform into binary variables.
`time`	Numeric. Time point where the AUC will be evaluated (default: NULL).
`type`	Character. Prediction type: "lp", "risk", "expected" or "survival" (default: "lp").
`method`	Character. Prediction method. It can be compute by using the cox model "cox" or by using W.star "W.star" (default: "cox").

Details

The function initiates by determining the prediction method specified by the user. If the "cox" method is chosen, the function computes the score matrix using the `predict.Coxmos` function. This score matrix serves as a foundation for subsequent predictions. It's imperative to note that for prediction types "expected" and "survival", a specific time point must be provided to ensure accurate predictions. The function then leverages the `predict` function from the Cox model to compute the desired prediction metric.

Alternatively, if the "W.star" method is selected, the function computes the prediction values based on the W* matrix and the Cox model's coefficients. This involves normalization of the input data, ensuring it aligns with the training data's distribution. The normalization process considers mean and standard deviation values from the model, ensuring consistency in predictions. The resultant prediction values are then computed as a linear combination of the normalized data and the derived coefficients.

It's worth noting that the function is meticulously designed to handle potential inconsistencies or missing components in the model, ensuring robustness in predictions and minimizing potential errors during execution.

Value

Return the "lp", "risk", "expected" or "survival" metric for test data using the specific Coxmos model.

Author(s)

Pedro Salguero Garcia. Maintainer: pedsalga@upv.edu.es

Examples

```
data("X_proteomic")
data("Y_proteomic")
set.seed(123)
index_train <- caret::createDataPartition(Y_proteomic$event, p = .5, list = FALSE, times = 1)
X_train <- X_proteomic[index_train,1:50]
Y_train <- Y_proteomic[index_train,]

X_test <- X_proteomic[-index_train,1:50]
Y_test <- Y_proteomic[-index_train,]

model_icox <- splsicox(X_train, Y_train, n.comp = 2)
cox.prediction(model = model_icox, new_data = X_test, type = "lp")
```

---

coxEN *coxEN*

Description

This function performs a cox elastic net model (based on glmnet R package). The function returns a Coxmos model with the attribute model as "coxEN".

Usage

```
coxEN(
  X,
  Y,
  EN.alpha = 0.5,
  max.variables = 15,
  x.center = TRUE,
  x.scale = FALSE,
  remove_near_zero_variance = TRUE,
  remove_zero_variance = FALSE,
  toKeep.zv = NULL,
  remove_non_significant = FALSE,
```

```
  alpha = 0.05,
  MIN_EPV = 5,
  returnData = TRUE,
  verbose = FALSE
)
```

Arguments

X	Numeric matrix or data.frame. Explanatory variables. Qualitative variables must be transform into binary variables.
Y	Numeric matrix or data.frame. Response variables. Object must have two columns named as "time" and "event". For event column, accepted values are: 0/1 or FALSE/TRUE for censored and event observations.
EN.alpha	Numeric. Elastic net mixing parameter. If EN.alpha = 1 is the lasso penalty, and EN.alpha = 0 the ridge penalty (default: 0.5). NOTE: When ridge penalty is used, EVP and max.variables will not be used.
max.variables	Numeric. Maximum number of variables you want to keep in the cox model. If MIN_EPV is not meet, the value will be change automatically (default: 20).
x.center	Logical. If x.center = TRUE, X matrix is centered to zero means (default: TRUE).
x.scale	Logical. If x.scale = TRUE, X matrix is scaled to unit variances (default: FALSE).
remove_near_zero_variance	Logical. If remove_near_zero_variance = TRUE, near zero variance variables will be removed (default: TRUE).
remove_zero_variance	Logical. If remove_zero_variance = TRUE, zero variance variables will be removed (default: TRUE).
toKeep.zv	Character vector. Name of variables in X to not be deleted by (near) zero variance filtering (default: NULL).
remove_non_significant	Logical. If remove_non_significant = TRUE, non-significant variables/components in final cox model will be removed until all variables are significant by forward selection (default: FALSE).
alpha	Numeric. Numerical values are regarded as significant if they fall below the threshold (default: 0.05).
MIN_EPV	Numeric. Minimum number of Events Per Variable (EPV) you want reach for the final cox model. Used to restrict the number of variables/components can be computed in final cox models. If the minimum is not meet, the model cannot be computed (default: 5).
returnData	Logical. Return original and normalized X and Y matrices (default: TRUE).
verbose	Logical. If verbose = TRUE, extra messages could be displayed (default: FALSE).

Details

The coxEN function is designed to handle survival data using the elastic net regularization. The function is particularly useful when dealing with high-dimensional datasets where the number of

predictors exceeds the number of observations. The elastic net regularization combines the strengths of both lasso and ridge regression. The `EN.alpha` parameter controls the balance between lasso and ridge penalties. It's important to note that when using the ridge penalty (`EN.alpha = 0`), the EVP and `max.variables` parameters will not be considered.

Value

Instance of class "Coxmos" and model "coxEN". The class contains the following elements: `X`: List of normalized X data information.

- `(data)`: normalized X matrix
- `(x.mean)`: mean values for X matrix
- `(x.sd)`: standard deviation for X matrix

`Y`: List of normalized Y data information.

- `(data)`: normalized X matrix
- `(y.mean)`: mean values for Y matrix
- `(y.sd)`: standard deviation for Y matrix'

`survival_model`: List of survival model information.

- `fit`: coxph object.
- `AIC`: AIC of cox model.
- `BIC`: BIC of cox model.
- `lp`: linear predictors for train data.
- `coef`: Coefficients for cox model.
- `YChapeau`: Y Chapeau residuals.
- `Yresidus`: Y residuals.

`opt.lambda`: Optimal lambda computed by the model with maximum % Var from glmnet function.

`EN.alpha`: EN.alpha selected

`n.var`: Number of variables selected

`call`: call function

`X_input`: X input matrix

`Y_input`: Y input matrix

`convergence_issue`: If any convergence issue has been found.

`alpha`: alpha value selected

`selected_variables_cox`: Variables selected to enter the cox model.

`nsv`: Variables removed by cox alpha cutoff.

`removed_variables_correlation`: Variables removed by being high correlated with other variables.

`nzv`: Variables removed by remove_near_zero_variance or remove_zero_variance.

`nz_coeffvar`: Variables removed by coefficient variation near zero.

`class`: Model class.

`time`: time consumed for running the cox analysis.

Author(s)

Pedro Salguero Garcia. Maintainer: pedsalga@upv.edu.es

References

Simon N, Friedman JH, Friedman JH, Hastie T, Tibshirani R (2011). "Regularization Paths for Cox's Proportional Hazards Model via Coordinate Descent." *Journal of Statistical Software*. doi:10.18637/jss.v039.i05, https://pubmed.ncbi.nlm.nih.gov/27065756/.

Examples

```
data("X_proteomic")
data("Y_proteomic")
X <- X_proteomic[,1:50]
Y <- Y_proteomic
coxEN(X, Y, EN.alpha = 0.75, x.center = TRUE, x.scale = TRUE, remove_non_significant = TRUE)
```

---

coxSW *coxSW*

Description

The coxSW function conducts a stepwise Cox regression analysis on survival data, leveraging the capabilities of the My.stepwise R package. The primary objective of this function is to identify the most significant predictors for survival data by iteratively adding or removing predictors based on their statistical significance in the model. The resulting model is of class "Coxmos" with an attribute model labeled as "coxSW".

Usage

```
coxSW(
  X,
  Y,
  max.variables = 20,
  BACKWARDS = TRUE,
  alpha_ENT = 0.1,
  alpha_OUT = 0.15,
  toKeep.sw = NULL,
  initialModel = NULL,
  x.center = TRUE,
  x.scale = FALSE,
  remove_near_zero_variance = TRUE,
  remove_zero_variance = FALSE,
  toKeep.zv = NULL,
  remove_non_significant = FALSE,
  alpha = 0.05,
  MIN_EPV = 5,
```

```
  returnData = TRUE,
  verbose = FALSE
)
```

Arguments

X	Numeric matrix or data.frame. Explanatory variables. Qualitative variables must be transform into binary variables.
Y	Numeric matrix or data.frame. Response variables. Object must have two columns named as "time" and "event". For event column, accepted values are: 0/1 or FALSE/TRUE for censored and event observations.
max.variables	Numeric. Maximum number of variables you want to keep in the cox model. If MIN_EPV is not meet, the value will be change automatically (default: 20).
BACKWARDS	Logical. If BACKWARDS = TRUE, backward strategy is performed (default: TRUE).
alpha_ENT	Numeric. Maximum P-Value for a variable to enter the model (default: 0.10).
alpha_OUT	Numeric. Minimum P-Value for a variable to leave the model (default: 0.15).
toKeep.sw	Character vector. Name of variables in X to not be deleted by Step-wise selection (default: NULL).
initialModel	Character vector. Name of variables in X to include in the initial model (default: NULL).
x.center	Logical. If x.center = TRUE, X matrix is centered to zero means (default: TRUE).
x.scale	Logical. If x.scale = TRUE, X matrix is scaled to unit variances (default: FALSE).
remove_near_zero_variance	Logical. If remove_near_zero_variance = TRUE, near zero variance variables will be removed (default: TRUE).
remove_zero_variance	Logical. If remove_zero_variance = TRUE, zero variance variables will be removed (default: TRUE).
toKeep.zv	Character vector. Name of variables in X to not be deleted by (near) zero variance filtering (default: NULL).
remove_non_significant	Logical. If remove_non_significant = TRUE, non-significant variables/components in final cox model will be removed until all variables are significant by forward selection (default: FALSE).
alpha	Numeric. Numerical values are regarded as significant if they fall below the threshold (default: 0.05).
MIN_EPV	Numeric. Minimum number of Events Per Variable (EPV) you want reach for the final cox model. Used to restrict the number of variables/components can be computed in final cox models. If the minimum is not meet, the model cannot be computed (default: 5).
returnData	Logical. Return original and normalized X and Y matrices (default: TRUE).
verbose	Logical. If verbose = TRUE, extra messages could be displayed (default: FALSE).

Details

The coxSW function employs a stepwise regression technique tailored for survival data. This method is particularly beneficial when dealing with a plethora of predictors, and there's a necessity to distill the model to its most impactful variables. The stepwise procedure can be configured to operate in forward, backward, or a hybrid mode, contingent on the parameters specified by the user.

During the iterative process, variables are evaluated for inclusion or exclusion based on predefined significance levels (alpha_ENT for entry and alpha_OUT for removal). This ensures that the model retains only those predictors that meet the significance criteria, thereby enhancing the model's interpretability and predictive power.

Additionally, the function offers several preprocessing options, such as centering and scaling of the predictor matrix, removal of variables with near-zero or zero variance, and the ability to enforce the inclusion of specific variables in the model. These preprocessing steps are crucial for ensuring the robustness and stability of the resulting Cox regression model.

It's worth noting that the function is equipped to handle both numeric and binary categorical predictors. However, it's imperative that categorical variables are appropriately transformed into binary format before analysis. The outcome or response variable should comprise two columns: "time" representing the survival time and "event" indicating the occurrence of the event of interest.

Value

Instance of class "Coxmos" and model "coxSW". The class contains the following elements:

X: List of normalized X data information.

- (data): normalized X matrix
- (x.mean): mean values for X matrix
- (x.sd): standard deviation for X matrix

Y: List of normalized Y data information.

- (data): normalized Y matrix
- (y.mean): mean values for Y matrix
- (y.sd): standard deviation for Y matrix

survival_model: List of survival model information

- fit: coxph object.
- AIC: AIC of cox model.
- BIC: BIC of cox model.
- lp: linear predictors for train data.
- coef: Coefficients for cox model.
- YChapeau: Y Chapeau residuals.
- Yresidus: Y residuals.

`call`: call function

`X_input`: X input matrix

`Y_input`: Y input matrix

`nsv`: Variables removed by remove_non_significant if any.

`nzv`: Variables removed by remove_near_zero_variance or remove_zero_variance.

`nz_coeffvar`: Variables removed by coefficient variation near zero.

`removed_variables_correlation`: Variables removed by being high correlated with other variables.

`class`: Model class.

`time`: time consumed for running the cox analysis.

Author(s)

Pedro Salguero Garcia. Maintainer: pedsalga@upv.edu.es

References

Efroymson MA (1960). "Multiple Regression Analysis." *Mathematical Methods for Digital Computers*. Company ISC (2017). "My.stepwise: Stepwise Variable Selection Procedures for Regression Analysis." https://cran.r-project.org/package=My.stepwise.

Examples

```
data("X_proteomic")
data("Y_proteomic")
X <- X_proteomic[,1:10]
Y <- Y_proteomic
coxSW(X, Y, x.center = TRUE, x.scale = TRUE)
```

CV *The cross-validation bandwidth selection for weighted data*

Description

This function computes the data-driven bandwidth for smoothing the ROC (or distribution) function using the CV method of Beyene and El Ghouch (2020). This is an extension of the classical (unweighted) cross-validation bandwith selection method to the case of weighted data.

Usage

```
CV(X, wt, ktype = "normal")
```

Arguments

X	The numeric data vector.
wt	The non-negative weight vector.
ktype	A character string giving the type kernel to be used: "normal", "epanechnikov", "biweight", or "triweight". By default, the "normal" kernel is used.

Details

Bowman et al (1998) proposed the cross-validation bandwidth selection method for unweighted kernal smoothed distribution function. This method is implemented in the R package kerdiest. We adapted this for the case of weighted data by incorporating the weight variable into the cross-validation function of Bowman's method. See Beyene and El Ghouch (2020) for details.

Value

Returns the computed value for the bandwith parameter.

Author(s)

Kassu Mehari Beyene, Catholic University of Louvain. <kasu.beyene@uclouvain.be>

Anouar El Ghouch, Catholic University of Louvain. <anouar.elghouch@uclouvain.be>

References

Beyene, K. M. and El Ghouch A. (2020). Smoothed time-dependent ROC curves for right-censored survival data. *submitted*.

Bowman A., Hall P. and Trvan T.(1998). Bandwidth selection for the smoothing of distribution functions. *Biometrika* 85:799-808.

Quintela-del-Rio, A. and Estevez-Perez, G. (2015). kerdiest: Nonparametric kernel estimation of the distribution function, bandwidth selection and estimation of related functions. R package version 1.2.

---

cv.coxEN *coxEN Cross-Validation*

Description

This function performs cross-validated CoxEN (coxEN). The function returns the optimal number of EN penalty value based on cross-validation. The performance could be based on multiple metrics as Area Under the Curve (AUC), Brier Score or C-Index. Furthermore, the user could establish more than one metric simultaneously.

Usage

```
cv.coxEN(
  X,
  Y,
  EN.alpha.list = seq(0, 1, 0.1),
  max.variables = 15,
  n_run = 3,
  k_folds = 10,
  x.center = TRUE,
  x.scale = FALSE,
  remove_near_zero_variance = TRUE,
  remove_zero_variance = TRUE,
  toKeep.zv = NULL,
  remove_variance_at_fold_level = FALSE,
  remove_non_significant_models = FALSE,
  remove_non_significant = FALSE,
  alpha = 0.05,
  w_AIC = 0,
  w_c.index = 0,
  w_AUC = 1,
  w_BRIER = 0,
  times = NULL,
  max_time_points = 15,
  MIN_AUC_INCREASE = 0.01,
  MIN_AUC = 0.8,
  MIN_COMP_TO_CHECK = 3,
  pred.attr = "mean",
  pred.method = "cenROC",
  fast_mode = FALSE,
  MIN_EPV = 5,
  return_models = FALSE,
  returnData = FALSE,
  PARALLEL = FALSE,
  verbose = FALSE,
  seed = 123
)
```

Arguments

X	Numeric matrix or data.frame. Explanatory variables. Qualitative variables must be transform into binary variables.
Y	Numeric matrix or data.frame. Response variables. Object must have two columns named as "time" and "event". For event column, accepted values are: 0/1 or FALSE/TRUE for censored and event observations.
EN.alpha.list	Numeric vector. Elastic net mixing parameter values to test in cross validation. EN.alpha = 1 is the lasso penalty, and EN.alpha = 0 the ridge penalty (default: seq(0,1,0.1)).

`max.variables` Numeric. Maximum number of variables you want to keep in the cox model. If MIN_EPV is not meet, the value will be change automatically (default: 20).

`n_run` Numeric. Number of runs for cross validation (default: 3).

`k_folds` Numeric. Number of folds for cross validation (default: 10).

`x.center` Logical. If x.center = TRUE, X matrix is centered to zero means (default: TRUE).

`x.scale` Logical. If x.scale = TRUE, X matrix is scaled to unit variances (default: FALSE).

`remove_near_zero_variance`
Logical. If remove_near_zero_variance = TRUE, near zero variance variables will be removed (default: TRUE).

`remove_zero_variance`
Logical. If remove_zero_variance = TRUE, zero variance variables will be removed (default: TRUE).

`toKeep.zv` Character vector. Name of variables in X to not be deleted by (near) zero variance filtering (default: NULL).

`remove_variance_at_fold_level`
Logical. If remove_variance_at_fold_level = TRUE, (near) zero variance will be removed at fold level (default: FALSE).

`remove_non_significant_models`
Logical. If remove_non_significant_models = TRUE, non-significant models are removed before computing the evaluation. A non-significant model is a model with at least one component/variable with a P-Value higher than the alpha cutoff.

`remove_non_significant`
Logical. If remove_non_significant = TRUE, non-significant variables/components in final cox model will be removed until all variables are significant by forward selection (default: FALSE).

`alpha` Numeric. Numerical values are regarded as significant if they fall below the threshold (default: 0.05).

`w_AIC` Numeric. Weight for AIC evaluator. All weights must sum 1 (default: 0).

`w_c.index` Numeric. Weight for C-Index evaluator. All weights must sum 1 (default: 0).

`w_AUC` Numeric. Weight for AUC evaluator. All weights must sum 1 (default: 1).

`w_BRIER` Numeric. Weight for BRIER SCORE evaluator. All weights must sum 1 (default: 0).

`times` Numeric vector. Time points where the AUC will be evaluated. If NULL, a maximum of 'max_time_points' points will be selected equally distributed (default: NULL).

`max_time_points`
Numeric. Maximum number of time points to use for evaluating the model (default: 15).

`MIN_AUC_INCREASE`
Numeric. Minimum improvement between different cross validation models to continue evaluating higher values in the multiple tested parameters. If it is not reached for next 'MIN_COMP_TO_CHECK' models and the minimum 'MIN_AUC' value is reached, the evaluation stops (default: 0.01).

MIN_AUC — Numeric. Minimum AUC desire to reach cross-validation models. If the minimum is reached, the evaluation could stop if the improvement does not reach an AUC higher than adding the 'MIN_AUC_INCREASE' value (default: 0.8).

MIN_COMP_TO_CHECK — Numeric. Number of penalties/components to evaluate to check if the AUC improves. If for the next 'MIN_COMP_TO_CHECK' the AUC is not better and the 'MIN_AUC' is meet, the evaluation could stop (default: 3).

pred.attr — Character. Way to evaluate the metric selected. Must be one of the following: "mean" or "median" (default: "mean").

pred.method — Character. AUC evaluation algorithm method for evaluate the model performance. Must be one of the following: "risksetROC", "survivalROC", "cenROC", "nsROC", "smoothROCtime_C", "smoothROCtime_I" (default: "cenROC").

fast_mode — Logical. If fast_mode = TRUE, for each run, only one fold is evaluated simultaneously. If fast_mode = FALSE, for each run, all linear predictors are computed for test observations. Once all have their linear predictors, the evaluation is perform across all the observations together (default: FALSE).

MIN_EPV — Numeric. Minimum number of Events Per Variable (EPV) you want reach for the final cox model. Used to restrict the number of variables/components can be computed in final cox models. If the minimum is not meet, the model cannot be computed (default: 5).

return_models — Logical. Return all models computed in cross validation (default: FALSE).

returnData — Logical. Return original and normalized X and Y matrices (default: TRUE).

PARALLEL — Logical. Run the cross validation with multicore option. As many cores as your total cores - 1 will be used. It could lead to higher RAM consumption (default: FALSE).

verbose — Logical. If verbose = TRUE, extra messages could be displayed (default: FALSE).

seed — Number. Seed value for performing runs/folds divisions (default: 123).

Details

The coxEN Cross-Validation function provides a robust mechanism to optimize the hyperparameters of the cox elastic net model through cross-validation. By systematically evaluating a range of elastic net mixing parameters (EN.alpha.list), this function identifies the optimal balance between lasso and ridge penalties for survival analysis.

The cross-validation process is structured across multiple runs (n_run) and folds (k_folds), ensuring a comprehensive assessment of model performance. Users can prioritize specific evaluation metrics, such as AUC, Brier Score, or C-Index, by assigning weights (w_AIC, w_c.index, w_AUC, w_BRIER). The function also offers flexibility in the AUC evaluation method (pred.method) and the attribute for metric evaluation (pred.attr).

One of the distinguishing features of this function is its adaptive evaluation process. The function can terminate the cross-validation early if the improvement in AUC does not exceed the MIN_AUC_INCREASE threshold or if a predefined AUC (MIN_AUC) is achieved. This adaptive approach ensures computational efficiency without compromising the quality of the results.

Data preprocessing options are integrated into the function, emphasizing the significance of data quality. Options to remove near-zero and zero variance variables, either globally or at the fold level, are available. The function also supports multicore processing (PARALLEL option) to expedite the cross-validation process.

Upon execution, the function returns a detailed output, encompassing information about the best model, performance metrics at various granularities (fold, run, component), and if desired, all cross-validated models.

Value

Instance of class "Coxmos" and model "cv.coxEN". The class contains the following elements: best_model_info: A data.frame with the information for the best model. df_results_folds: A data.frame with fold-level information. df_results_runs: A data.frame with run-level information. df_results_comps: A data.frame with component-level information (for cv.coxEN, EN.alpha information).

lst_models: If return_models = TRUE, return a the list of all cross-validated models. pred.method: AUC evaluation algorithm method for evaluate the model performance.

opt.EN.alpha: Optimal EN.alpha value selected by the best_model. opt.nvar: Optimal number of variables selected by the best_model.

plot_AIC: AIC plot by each hyper-parameter. plot_c_index: C-Index plot by each hyper-parameter. plot_BRIER: Brier Score plot by each hyper-parameter. plot_AUC: AUC plot by each hyper-parameter.

class: Cross-Validated model class.

lst_train_indexes: List (of lists) of indexes for the observations used in each run/fold for train the models. lst_test_indexes: List (of lists) of indexes for the observations used in each run/fold for test the models.

time: time consumed for running the cross-validated function.

Author(s)

Pedro Salguero Garcia. Maintainer: pedsalga@upv.edu.es

Examples

```
data("X_proteomic")
data("Y_proteomic")
set.seed(123)
index_train <- caret::createDataPartition(Y_proteomic$event, p = .5, list = FALSE, times = 1)
X_train <- X_proteomic[index_train,1:50]
Y_train <- Y_proteomic[index_train,]
cv.coxEN_model <- cv.coxEN(X_train, Y_train, EN.alpha.list = c(0.1,0.5),
x.center = TRUE, x.scale = TRUE)
```

`cv.isb.splsdrcox` *Cross validation cv.isb.splsdrcox*

Description

This function performs cross-validated sparse partial least squares iterative single block for splsdrcox. The function returns the optimal number of components and the optimal sparsity penalty value based on cross-validation. The performance could be based on multiple metrics as Area Under the Curve (AUC), Brier Score or C-Index. Furthermore, the user could establish more than one metric simultaneously.

Usage

```
cv.isb.splsdrcox(
  X,
  Y,
  max.ncomp = 8,
  penalty.list = seq(0.1, 0.9, 0.2),
  n_run = 3,
  k_folds = 10,
  x.center = TRUE,
  x.scale = FALSE,
  remove_near_zero_variance = TRUE,
  remove_zero_variance = TRUE,
  toKeep.zv = NULL,
  remove_variance_at_fold_level = FALSE,
  remove_non_significant_models = FALSE,
  remove_non_significant = FALSE,
  alpha = 0.05,
  w_AIC = 0,
  w_c.index = 0,
  w_AUC = 1,
  w_BRIER = 0,
  times = NULL,
  max_time_points = 15,
  MIN_AUC_INCREASE = 0.01,
  MIN_AUC = 0.8,
  MIN_COMP_TO_CHECK = 3,
  pred.attr = "mean",
  pred.method = "cenROC",
  fast_mode = FALSE,
  MIN_EPV = 5,
  returnData = TRUE,
  return_models = FALSE,
  PARALLEL = FALSE,
  verbose = FALSE,
  seed = 123
```

)

Arguments

X	Numeric matrix or data.frame. Explanatory variables. Qualitative variables must be transform into binary variables.
Y	Numeric matrix or data.frame. Response variables. Object must have two columns named as "time" and "event". For event column, accepted values are: 0/1 or FALSE/TRUE for censored and event observations.
max.ncomp	Numeric. Maximum number of PLS components to compute for the cross validation (default: 8).
penalty.list	Numeric vector. Vector of penalty values. Penalty for sPLS-DRCOX. If penalty = 0 no penalty is applied, when penalty = 1 maximum penalty (no variables are selected) based on 'plsRcox' penalty. Equal or greater than 1 cannot be selected (default: seq(0.1,0.9,0.2)).
n_run	Numeric. Number of runs for cross validation (default: 3).
k_folds	Numeric. Number of folds for cross validation (default: 10).
x.center	Logical. If x.center = TRUE, X matrix is centered to zero means (default: TRUE).
x.scale	Logical. If x.scale = TRUE, X matrix is scaled to unit variances (default: FALSE).
remove_near_zero_variance	Logical. If remove_near_zero_variance = TRUE, near zero variance variables will be removed (default: TRUE).
remove_zero_variance	Logical. If remove_zero_variance = TRUE, zero variance variables will be removed (default: TRUE).
toKeep.zv	Character vector. Name of variables in X to not be deleted by (near) zero variance filtering (default: NULL).
remove_variance_at_fold_level	Logical. If remove_variance_at_fold_level = TRUE, (near) zero variance will be removed at fold level (default: FALSE).
remove_non_significant_models	Logical. If remove_non_significant_models = TRUE, non-significant models are removed before computing the evaluation. A non-significant model is a model with at least one component/variable with a P-Value higher than the alpha cutoff.
remove_non_significant	Logical. If remove_non_significant = TRUE, non-significant variables/components in final cox model will be removed until all variables are significant by forward selection (default: FALSE).
alpha	Numeric. Numerical values are regarded as significant if they fall below the threshold (default: 0.05).
w_AIC	Numeric. Weight for AIC evaluator. All weights must sum 1 (default: 0).
w_c.index	Numeric. Weight for C-Index evaluator. All weights must sum 1 (default: 0).

w_AUC Numeric. Weight for AUC evaluator. All weights must sum 1 (default: 1).

w_BRIER Numeric. Weight for BRIER SCORE evaluator. All weights must sum 1 (default: 0).

times Numeric vector. Time points where the AUC will be evaluated. If NULL, a maximum of 'max_time_points' points will be selected equally distributed (default: NULL).

max_time_points
Numeric. Maximum number of time points to use for evaluating the model (default: 15).

MIN_AUC_INCREASE
Numeric. Minimum improvement between different cross validation models to continue evaluating higher values in the multiple tested parameters. If it is not reached for next 'MIN_COMP_TO_CHECK' models and the minimum 'MIN_AUC' value is reached, the evaluation stops (default: 0.01).

MIN_AUC Numeric. Minimum AUC desire to reach cross-validation models. If the minimum is reached, the evaluation could stop if the improvement does not reach an AUC higher than adding the 'MIN_AUC_INCREASE' value (default: 0.8).

MIN_COMP_TO_CHECK
Numeric. Number of penalties/components to evaluate to check if the AUC improves. If for the next 'MIN_COMP_TO_CHECK' the AUC is not better and the 'MIN_AUC' is meet, the evaluation could stop (default: 3).

pred.attr Character. Way to evaluate the metric selected. Must be one of the following: "mean" or "median" (default: "mean").

pred.method Character. AUC evaluation algorithm method for evaluate the model performance. Must be one of the following: "risksetROC", "survivalROC", "cenROC", "nsROC", "smoothROCtime_C", "smoothROCtime_I" (default: "cenROC").

fast_mode Logical. If fast_mode = TRUE, for each run, only one fold is evaluated simultaneously. If fast_mode = FALSE, for each run, all linear predictors are computed for test observations. Once all have their linear predictors, the evaluation is perform across all the observations together (default: FALSE).

MIN_EPV Numeric. Minimum number of Events Per Variable (EPV) you want reach for the final cox model. Used to restrict the number of variables/components can be computed in final cox models. If the minimum is not meet, the model cannot be computed (default: 5).

returnData Logical. Return original and normalized X and Y matrices (default: TRUE).

return_models Logical. Return all models computed in cross validation (default: FALSE).

PARALLEL Logical. Run the cross validation with multicore option. As many cores as your total cores - 1 will be used. It could lead to higher RAM consumption (default: FALSE).

verbose Logical. If verbose = TRUE, extra messages could be displayed (default: FALSE).

seed Number. Seed value for performing runs/folds divisions (default: 123).

Details

The `cv.isb.splsdrcox` function performs cross-validation for the integrative single-block sparse partial least squares deviance residual Cox analysis. Unlike the single-block (SB) approach, the integrative single-block (ISB) method allows for the consideration of multiple blocks of data, potentially from different sources or types, to be integrated into a single model. A key distinction of the ISB approach is its ability to compute and optimize hyperparameters individually for each block, rather than applying a uniform set of hyperparameters across all blocks. This ensures that each block's unique characteristics are taken into account, leading to a more tailored and potentially more accurate model.

Cross-validation is essential for assessing the generalizability of the model and avoiding overfitting. By partitioning the original dataset into training and test sets multiple times, the function evaluates the model's performance across different subsets of the data. This iterative process ensures that the model's performance is robust and not overly reliant on a specific partition of the data.

The function evaluates a range of hyperparameters, including the number of latent components (`max.ncomp`) and the penalty for variable selection (`penalty.list`). For each combination of hyperparameters, the dataset is divided into training and test sets based on the specified number of folds (`k_folds`). The model is then trained on the training set and its performance is assessed on the test set. This process is repeated for the specified number of runs (`n_run`), providing a comprehensive evaluation of the model's performance.

Various evaluation metrics, such as AIC, C-Index, Brier Score, and AUC, are computed for each combination of hyperparameters. These metrics provide insights into the model's accuracy, discriminative ability, and calibration. The function then identifies the optimal hyperparameters that yield the best performance based on these metrics.

In summary, the `cv.isb.splsdrcox` function offers a robust and integrative approach for hyperparameter tuning and model evaluation for the sparse partial least squares deviance residual Cox analysis. By allowing individualized hyperparameter optimization for each block, the ISB approach ensures a more nuanced and potentially more accurate model compared to the traditional SB method.

Value

Instance of class "Coxmos" and model "sb.splscox". The class contains the following elements: `X`: List of normalized X data information.

- `(data)`: normalized X matrix
- `(weightings)`: PLS weights
- `(weightings_norm)`: PLS normalize weights
- `(W.star)`: PLS W* vector
- `(scores)`: PLS scores/variates
- `(x.mean)`: mean values for X matrix
- `(x.sd)`: standard deviation for X matrix

`Y`: List of normalized Y data information.

- `(deviance_residuals)`: deviance residual vector used as Y matrix in the sPLS.
- `(dr.mean)`: mean values for deviance residuals Y matrix

- (dr.sd): standard deviation for deviance residuals Y matrix'
- (data): normalized X matrix
- (y.mean): mean values for Y matrix
- (y.sd): standard deviation for Y matrix'

survival_model: List of survival model information.

- fit: coxph object.
- AIC: AIC of cox model.
- BIC: BIC of cox model.
- lp: linear predictors for train data.
- coef: Coefficients for cox model.
- YChapeau: Y Chapeau residuals.
- Yresidus: Y residuals.

list_spls_models: List of sPLS-DRCOX models computed for each block.

n.comp: Number of components selected.

penalty Penalty applied.

call: call function

X_input: X input matrix

Y_input: Y input matrix

nzv: Variables removed by remove_near_zero_variance or remove_zero_variance.

nz_coeffvar: Variables removed by coefficient variation near zero.

class: Model class.

time: time consumed for running the cox analysis.

Author(s)

Pedro Salguero Garcia. Maintainer: pedsalga@upv.edu.es

Examples

```
data("X_multiomic")
data("Y_multiomic")
set.seed(123)
index_train <- caret::createDataPartition(Y_multiomic$event, p = .5, list = FALSE, times = 1)
X_train <- X_multiomic
X_train$mirna <- X_train$mirna[index_train,1:50]
X_train$proteomic <- X_train$proteomic[index_train,1:50]
Y_train <- Y_multiomic[index_train,]
isb.splsdrcox_model <- cv.isb.splsdrcox(X_train, Y_train, max.ncomp = 2, penalty.list = c(0.5),
n_run = 1, k_folds = 2, x.center = TRUE, x.scale = TRUE)
```

`cv.isb.splsicox` *Cross validation cv.isb.splsicox*

Description

This function performs cross-validated sparse partial least squares iterative single block for splsicox. The function returns the optimal number of components and the optimal sparsity penalty value based on cross-validation. The performance could be based on multiple metrics as Area Under the Curve (AUC), Brier Score or C-Index. Furthermore, the user could establish more than one metric simultaneously.

Usage

```
cv.isb.splsicox(
  X,
  Y,
  max.ncomp = 8,
  penalty.list = seq(0.1, 0.9, 0.2),
  n_run = 3,
  k_folds = 10,
  x.center = TRUE,
  x.scale = FALSE,
  remove_near_zero_variance = TRUE,
  remove_zero_variance = TRUE,
  toKeep.zv = NULL,
  remove_variance_at_fold_level = FALSE,
  remove_non_significant_models = FALSE,
  remove_non_significant = FALSE,
  alpha = 0.05,
  w_AIC = 0,
  w_c.index = 0,
  w_AUC = 1,
  w_BRIER = 0,
  times = NULL,
  max_time_points = 15,
  MIN_AUC_INCREASE = 0.01,
  MIN_AUC = 0.8,
  MIN_COMP_TO_CHECK = 3,
  pred.attr = "mean",
  pred.method = "cenROC",
  fast_mode = FALSE,
  MIN_EPV = 5,
  returnData = TRUE,
  return_models = FALSE,
  PARALLEL = FALSE,
  verbose = FALSE,
  seed = 123
```

)

Arguments

X	Numeric matrix or data.frame. Explanatory variables. Qualitative variables must be transform into binary variables.
Y	Numeric matrix or data.frame. Response variables. Object must have two columns named as "time" and "event". For event column, accepted values are: 0/1 or FALSE/TRUE for censored and event observations.
max.ncomp	Numeric. Maximum number of PLS components to compute for the cross validation (default: 8).
penalty.list	Numeric vector. Penalty for variable selection for the individual cox models. Variables with a lower P-Value than 1- "penalty" in the individual cox analysis will be keep for the sPLS-ICOX approach (default: seq(0.1,0.9,0.2)).
n_run	Numeric. Number of runs for cross validation (default: 3).
k_folds	Numeric. Number of folds for cross validation (default: 10).
x.center	Logical. If x.center = TRUE, X matrix is centered to zero means (default: TRUE).
x.scale	Logical. If x.scale = TRUE, X matrix is scaled to unit variances (default: FALSE).
remove_near_zero_variance	Logical. If remove_near_zero_variance = TRUE, near zero variance variables will be removed (default: TRUE).
remove_zero_variance	Logical. If remove_zero_variance = TRUE, zero variance variables will be removed (default: TRUE).
toKeep.zv	Character vector. Name of variables in X to not be deleted by (near) zero variance filtering (default: NULL).
remove_variance_at_fold_level	Logical. If remove_variance_at_fold_level = TRUE, (near) zero variance will be removed at fold level (default: FALSE).
remove_non_significant_models	Logical. If remove_non_significant_models = TRUE, non-significant models are removed before computing the evaluation.
remove_non_significant	Logical. If remove_non_significant = TRUE, non-significant variables/components in final cox model will be removed until all variables are significant by forward selection (default: FALSE).
alpha	Numeric. Numerical values are regarded as significant if they fall below the threshold (default: 0.05).
w_AIC	Numeric. Weight for AIC evaluator. All weights must sum 1 (default: 0).
w_c.index	Numeric. Weight for C-Index evaluator. All weights must sum 1 (default: 0).
w_AUC	Numeric. Weight for AUC evaluator. All weights must sum 1 (default: 1).
w_BRIER	Numeric. Weight for BRIER SCORE evaluator. All weights must sum 1 (default: 0).

`times` Numeric vector. Time points where the AUC will be evaluated. If NULL, a maximum of 'max_time_points' points will be selected equally distributed (default: NULL).

`max_time_points` Numeric. Maximum number of time points to use for evaluating the model (default: 15).

`MIN_AUC_INCREASE` Numeric. Minimum improvement between different cross validation models to continue evaluating higher values in the multiple tested parameters. If it is not reached for next 'MIN_COMP_TO_CHECK' models and the minimum 'MIN_AUC' value is reached, the evaluation stops (default: 0.01).

`MIN_AUC` Numeric. Minimum AUC desire to reach cross-validation models. If the minimum is reached, the evaluation could stop if the improvement does not reach an AUC higher than adding the 'MIN_AUC_INCREASE' value (default: 0.8).

`MIN_COMP_TO_CHECK` Numeric. Number of penalties/components to evaluate to check if the AUC improves. If for the next 'MIN_COMP_TO_CHECK' the AUC is not better and the 'MIN_AUC' is meet, the evaluation could stop (default: 3).

`pred.attr` Character. Way to evaluate the metric selected. Must be one of the following: "mean" or "median" (default: "mean").

`pred.method` Character. AUC evaluation algorithm method for evaluate the model performance. Must be one of the following: "risksetROC", "survivalROC", "cenROC", "nsROC", "smoothROCtime_C", "smoothROCtime_I" (default: "cenROC").

`fast_mode` Logical. If fast_mode = TRUE, for each run, only one fold is evaluated simultaneously. If fast_mode = FALSE, for each run, all linear predictors are computed for test observations. Once all have their linear predictors, the evaluation is perform across all the observations together (default: FALSE).

`MIN_EPV` Numeric. Minimum number of Events Per Variable (EPV) you want reach for the final cox model. Used to restrict the number of variables/components can be computed in final cox models. If the minimum is not meet, the model cannot be computed (default: 5).

`returnData` Logical. Return original and normalized X and Y matrices (default: TRUE).

`return_models` Logical. Return all models computed in cross validation (default: FALSE).

`PARALLEL` Logical. Run the cross validation with multicore option. As many cores as your total cores - 1 will be used. It could lead to higher RAM consumption (default: FALSE).

`verbose` Logical. If verbose = TRUE, extra messages could be displayed (default: FALSE).

`seed` Number. Seed value for performing runs/folds divisions (default: 123).

Details

The `cv.isb.splsicox` function performs cross-validation for the iterative single-block sparse partial least squares individual Cox analysis. Unlike the single-block (`sb`) approach, where each block is analyzed with the same number of components and penalties, the iterative single-block (`isb`)

approach allows for the specification of different numbers of components and penalties for each block. This provides a more tailored analysis for each block, recognizing that different blocks may have varying complexities and relationships with the outcome.

The function is designed to handle datasets with multiple blocks, processing each block individually in an iterative manner. This ensures a detailed examination of each block's contribution to the survival outcome without the interference of other blocks. This approach is distinct from multiblock methods where all blocks are analyzed simultaneously.

The cross-validation process involves partitioning the dataset into multiple subsets (folds) and then iteratively training the model on a subset of the data while validating it on the remaining data. This helps in determining the optimal hyperparameters for the model, such as the number of latent components and the penalty for variable selection.

Unlike the `sb` approach, which returns the optimal hyperparameters for further model training, the `isb` approach directly returns the final model. This model is constructed using the best-performing hyperparameters for each block, ensuring a more customized and potentially more accurate model.

The function offers flexibility in specifying various hyperparameters and options for data preprocessing. The output provides a comprehensive overview of the cross-validation results, including metrics like AIC, C-Index, Brier Score, and AUC for each hyper-parameter combination. Visualization tools are also provided to aid in understanding the model's performance across different hyperparameters.

Value

Instance of class "Coxmos" and model "sb.splsicox". The class contains the following elements: `X`: List of normalized X data information.

- `(data)`: normalized X matrix
- `(weightings)`: PLS weights
- `(weightings_norm)`: PLS normalize weights
- `(W.star)`: PLS W* vector
- `(scores)`: PLS scores/variates
- `(x.mean)`: mean values for X matrix
- `(x.sd)`: standard deviation for X matrix

`Y`: List of normalized Y data information.

- `(deviance_residuals)`: deviance residual vector used as Y matrix in the sPLS.
- `(dr.mean)`: mean values for deviance residuals Y matrix
- `(dr.sd)`: standard deviation for deviance residuals Y matrix'
- `(data)`: normalized X matrix
- `(y.mean)`: mean values for Y matrix
- `(y.sd)`: standard deviation for Y matrix'

`survival_model`: List of survival model information.

- `fit`: coxph object.

- AIC: AIC of cox model.
- BIC: BIC of cox model.
- lp: linear predictors for train data.
- coef: Coefficients for cox model.
- YChapeau: Y Chapeau residuals.
- Yresidus: Y residuals.

list_spls_models: List of sPLS-ICOX models computed for each block.

n.comp: Number of components selected.

penalty Penalty applied.

call: call function

X_input: X input matrix

Y_input: Y input matrix

nzv: Variables removed by remove_near_zero_variance or remove_zero_variance.

nz_coeffvar: Variables removed by coefficient variation near zero.

class: Model class.

time: time consumed for running the cox analysis.

Author(s)

Pedro Salguero Garcia. Maintainer: pedsalga@upv.edu.es

Examples

```
data("X_multiomic")
data("Y_multiomic")
set.seed(123)
index_train <- caret::createDataPartition(Y_multiomic$event, p = .5, list = FALSE, times = 1)
X_train <- X_multiomic
X_train$mirna <- X_train$mirna[index_train,1:20]
X_train$proteomic <- X_train$proteomic[index_train,1:20]
Y_train <- Y_multiomic[index_train,]
isb.splsicox_model <- cv.isb.splsicox(X_train, Y_train, max.ncomp = 1, penalty.list = c(0.5),
n_run = 1, k_folds = 2, x.center = TRUE, x.scale = TRUE)
```

---

cv.mb.splsdacox *MB.sPLS-DACOX Cross-Validation*

Description

The cv.mb.splsdacox function performs cross-validation for the MB.sPLS-DACOX model, a specialized model tailored for survival analysis with high-dimensional data. This function systematically evaluates the performance of the model across different hyperparameters and configurations to determine the optimal settings for the given data.

Usage

```
cv.mb.splsdacox(
  X,
  Y,
  max.ncomp = 8,
  vector = NULL,
  MIN_NVAR = 10,
  MAX_NVAR = 10000,
  n.cut_points = 5,
  EVAL_METHOD = "AUC",
  n_run = 3,
  k_folds = 10,
  x.center = TRUE,
  x.scale = FALSE,
  remove_near_zero_variance = TRUE,
  remove_zero_variance = TRUE,
  toKeep.zv = NULL,
  remove_variance_at_fold_level = FALSE,
  remove_non_significant_models = FALSE,
  remove_non_significant = FALSE,
  alpha = 0.05,
  w_AIC = 0,
  w_c.index = 0,
  w_AUC = 1,
  w_BRIER = 0,
  times = NULL,
  max_time_points = 15,
  MIN_AUC_INCREASE = 0.01,
  MIN_AUC = 0.8,
  MIN_COMP_TO_CHECK = 3,
  pred.attr = "mean",
  pred.method = "cenROC",
  max.iter = 200,
  fast_mode = FALSE,
  MIN_EPV = 5,
  return_models = FALSE,
  returnData = FALSE,
  PARALLEL = FALSE,
  verbose = FALSE,
  seed = 123
)
```

Arguments

X	Numeric matrix or data.frame. Explanatory variables. Qualitative variables must be transform into binary variables.
Y	Numeric matrix or data.frame. Response variables. Object must have two columns named as "time" and "event". For event column, accepted values are:

0/1 or FALSE/TRUE for censored and event observations.

`max.ncomp` Numeric. Maximum number of PLS components to compute for the cross validation (default: 8).

`vector` Numeric vector. Used for computing best number of variables. As many values as components have to be provided. If vector = NULL, an automatic detection is perform (default: NULL). If vector is a list, must be named as the names of X param followed by the number of variables to select.

`MIN_NVAR` Numeric. Minimum range size for computing cut points to select the best number of variables to use (default: 10).

`MAX_NVAR` Numeric. Maximum range size for computing cut points to select the best number of variables to use (default: 1000).

`n.cut_points` Numeric. Number of cut points for searching the optimal number of variables. If only two cut points are selected, minimum and maximum size are used (default: 5).

`EVAL_METHOD` Character. If EVAL_METHOD = "AUC", AUC metric will be use to compute the best number of variables. In other case, c-index metric will be used (default: "AUC").

`n_run` Numeric. Number of runs for cross validation (default: 3).

`k_folds` Numeric. Number of folds for cross validation (default: 10).

`x.center` Logical. If x.center = TRUE, X matrix is centered to zero means (default: TRUE).

`x.scale` Logical. If x.scale = TRUE, X matrix is scaled to unit variances (default: FALSE).

`remove_near_zero_variance`
Logical. If remove_near_zero_variance = TRUE, near zero variance variables will be removed (default: TRUE).

`remove_zero_variance`
Logical. If remove_zero_variance = TRUE, zero variance variables will be removed (default: TRUE).

`toKeep.zv` Character vector. Name of variables in X to not be deleted by (near) zero variance filtering (default: NULL).

`remove_variance_at_fold_level`
Logical. If remove_variance_at_fold_level = TRUE, (near) zero variance will be removed at fold level (default: FALSE).

`remove_non_significant_models`
Logical. If remove_non_significant_models = TRUE, non-significant models are removed before computing the evaluation. A non-significant model is a model with at least one component/variable with a P-Value higher than the alpha cutoff.

`remove_non_significant`
Logical. If remove_non_significant = TRUE, non-significant variables/components in final cox model will be removed until all variables are significant by forward selection (default: FALSE).

`alpha` Numeric. Numerical values are regarded as significant if they fall below the threshold (default: 0.05).

w_AIC Numeric. Weight for AIC evaluator. All weights must sum 1 (default: 0).

w_c.index Numeric. Weight for C-Index evaluator. All weights must sum 1 (default: 0).

w_AUC Numeric. Weight for AUC evaluator. All weights must sum 1 (default: 1).

w_BRIER Numeric. Weight for BRIER SCORE evaluator. All weights must sum 1 (default: 0).

times Numeric vector. Time points where the AUC will be evaluated. If NULL, a maximum of 'max_time_points' points will be selected equally distributed (default: NULL).

max_time_points
Numeric. Maximum number of time points to use for evaluating the model (default: 15).

MIN_AUC_INCREASE
Numeric. Minimum improvement between different cross validation models to continue evaluating higher values in the multiple tested parameters. If it is not reached for next 'MIN_COMP_TO_CHECK' models and the minimum 'MIN_AUC' value is reached, the evaluation stops (default: 0.01).

MIN_AUC Numeric. Minimum AUC desire to reach cross-validation models. If the minimum is reached, the evaluation could stop if the improvement does not reach an AUC higher than adding the 'MIN_AUC_INCREASE' value (default: 0.8).

MIN_COMP_TO_CHECK
Numeric. Number of penalties/components to evaluate to check if the AUC improves. If for the next 'MIN_COMP_TO_CHECK' the AUC is not better and the 'MIN_AUC' is meet, the evaluation could stop (default: 3).

pred.attr Character. Way to evaluate the metric selected. Must be one of the following: "mean" or "median" (default: "mean").

pred.method Character. AUC evaluation algorithm method for evaluate the model performance. Must be one of the following: "risksetROC", "survivalROC", "cenROC", "nsROC", "smoothROCtime_C", "smoothROCtime_I" (default: "cenROC").

max.iter Numeric. Maximum number of iterations for PLS convergence (default: 200).

fast_mode Logical. If fast_mode = TRUE, for each run, only one fold is evaluated simultaneously. If fast_mode = FALSE, for each run, all linear predictors are computed for test observations. Once all have their linear predictors, the evaluation is perform across all the observations together (default: FALSE).

MIN_EPV Numeric. Minimum number of Events Per Variable (EPV) you want reach for the final cox model. Used to restrict the number of variables/components can be computed in final cox models. If the minimum is not meet, the model cannot be computed (default: 5).

return_models Logical. Return all models computed in cross validation (default: FALSE).

returnData Logical. Return original and normalized X and Y matrices (default: TRUE).

PARALLEL Logical. Run the cross validation with multicore option. As many cores as your total cores - 1 will be used. It could lead to higher RAM consumption (default: FALSE).

verbose Logical. If verbose = TRUE, extra messages could be displayed (default: FALSE).

seed Number. Seed value for performing runs/folds divisions (default: 123).

Details

The function operates by partitioning the data into multiple subsets (folds) and iteratively holding out one subset for validation while training on the remaining subsets. The cross-validation process is repeated for a specified number of runs, ensuring a robust assessment of the model's performance. The function offers flexibility in terms of the number of PLS components, the range of variables considered, and the evaluation metrics used.

The function provides an option to center and scale the explanatory variables, which can be crucial for ensuring consistent performance, especially when the variables are measured on different scales. Additionally, the function incorporates features to handle near-zero and zero variance variables, which can be problematic in high-dimensional datasets.

For model evaluation, users can choose between various metrics, including AUC, c-index, and Brier Score. The function also allows for the specification of weights for these metrics, enabling users to prioritize certain metrics over others based on the research context.

The function's design also emphasizes computational efficiency. It offers a parallel processing option to expedite the cross-validation process, especially beneficial for large datasets. However, users should be cautious about potential high RAM consumption when using this option.

Value

Instance of class "Coxmos" and model "cv.MB.sPLS-DACOX". `best_model_info`: A data.frame with the information for the best model. `df_results_folds`: A data.frame with fold-level information. `df_results_runs`: A data.frame with run-level information. `df_results_comps`: A data.frame with component-level information (for cv.coxEN, EN.alpha information).

`lst_models`: If return_models = TRUE, return a the list of all cross-validated models. `pred.method`: AUC evaluation algorithm method for evaluate the model performance.

`opt.comp`: Optimal component selected by the best_model. `opt.nvar`: Optimal number of variables selected by the best_model.

`plot_AIC`: AIC plot by each hyper-parameter. `plot_c_index`: C-Index plot by each hyper-parameter. `plot_BRIER`: Brier Score plot by each hyper-parameter. `plot_AUC`: AUC plot by each hyper-parameter.

`class`: Cross-Validated model class.

`lst_train_indexes`: List (of lists) of indexes for the observations used in each run/fold for train the models. `lst_test_indexes`: List (of lists) of indexes for the observations used in each run/fold for test the models.

`time`: time consumed for running the cross-validated function.

Author(s)

Pedro Salguero Garcia. Maintainer: pedsalga@upv.edu.es

Examples

```
data("X_multiomic")
data("Y_multiomic")
set.seed(123)
```

```
index_train <- caret::createDataPartition(Y_multiomic$event, p = .5, list = FALSE, times = 1)
X_train <- X_multiomic
X_train$mirna <- X_train$mirna[index_train,1:50]
X_train$proteomic <- X_train$proteomic[index_train,1:50]
Y_train <- Y_multiomic[index_train,]
cv.mb.splsdacox_model <- cv.mb.splsdacox(X_train, Y_train, max.ncomp = 2, vector = NULL,
n_run = 1, k_folds = 2, x.center = TRUE, x.scale = TRUE)
```

`cv.mb.splsdrcox` *MB.sPLS-DRCOX Cross-Validation*

Description

The cv.mb.splsdrcox function performs cross-validation for the MB.sPLS-DRCOX model, a specialized model for survival analysis with high-dimensional data. This function systematically evaluates the performance of the model across different hyperparameters and configurations to determine the optimal settings for the given data.

Usage

```
cv.mb.splsdrcox(
  X,
  Y,
  max.ncomp = 8,
  vector = NULL,
  MIN_NVAR = 10,
  MAX_NVAR = 10000,
  n.cut_points = 5,
  EVAL_METHOD = "AUC",
  n_run = 3,
  k_folds = 10,
  x.center = TRUE,
  x.scale = FALSE,
  remove_near_zero_variance = TRUE,
  remove_zero_variance = TRUE,
  toKeep.zv = NULL,
  remove_variance_at_fold_level = FALSE,
  remove_non_significant_models = FALSE,
  remove_non_significant = FALSE,
  alpha = 0.05,
  w_AIC = 0,
  w_c.index = 0,
  w_AUC = 1,
  w_BRIER = 0,
  times = NULL,
  max_time_points = 15,
  MIN_AUC_INCREASE = 0.01,
```

```
  MIN_AUC = 0.8,
  MIN_COMP_TO_CHECK = 3,
  pred.attr = "mean",
  pred.method = "cenROC",
  max.iter = 200,
  fast_mode = FALSE,
  MIN_EPV = 5,
  return_models = FALSE,
  returnData = FALSE,
  PARALLEL = FALSE,
  verbose = FALSE,
  seed = 123
)
```

Arguments

X	Numeric matrix or data.frame. Explanatory variables. Qualitative variables must be transform into binary variables.
Y	Numeric matrix or data.frame. Response variables. Object must have two columns named as "time" and "event". For event column, accepted values are: 0/1 or FALSE/TRUE for censored and event observations.
max.ncomp	Numeric. Maximum number of PLS components to compute for the cross validation (default: 8).
vector	Numeric vector. Used for computing best number of variables. As many values as components have to be provided. If vector = NULL, an automatic detection is perform (default: NULL). If vector is a list, must be named as the names of X param followed by the number of variables to select.
MIN_NVAR	Numeric. Minimum range size for computing cut points to select the best number of variables to use (default: 10).
MAX_NVAR	Numeric. Maximum range size for computing cut points to select the best number of variables to use (default: 1000).
n.cut_points	Numeric. Number of cut points for searching the optimal number of variables. If only two cut points are selected, minimum and maximum size are used. For MB approaches as many as n.cut_points^n.blocks models will be computed as minimum (default: 5).
EVAL_METHOD	Character. If EVAL_METHOD = "AUC", AUC metric will be use to compute the best number of variables. In other case, c-index metric will be used (default: "AUC").
n_run	Numeric. Number of runs for cross validation (default: 3).
k_folds	Numeric. Number of folds for cross validation (default: 10).
x.center	Logical. If x.center = TRUE, X matrix is centered to zero means (default: TRUE).
x.scale	Logical. If x.scale = TRUE, X matrix is scaled to unit variances (default: FALSE).
remove_near_zero_variance	Logical. If remove_near_zero_variance = TRUE, near zero variance variables will be removed (default: TRUE).

`remove_zero_variance`
: Logical. If remove_zero_variance = TRUE, zero variance variables will be removed (default: TRUE).

`toKeep.zv`
: Character vector. Name of variables in X to not be deleted by (near) zero variance filtering (default: NULL).

`remove_variance_at_fold_level`
: Logical. If remove_variance_at_fold_level = TRUE, (near) zero variance will be removed at fold level (default: FALSE).

`remove_non_significant_models`
: Logical. If remove_non_significant_models = TRUE, non-significant models are removed before computing the evaluation. A non-significant model is a model with at least one component/variable with a P-Value higher than the alpha cutoff.

`remove_non_significant`
: Logical. If remove_non_significant = TRUE, non-significant variables/components in final cox model will be removed until all variables are significant by forward selection (default: FALSE).

`alpha`
: Numeric. Numerical values are regarded as significant if they fall below the threshold (default: 0.05).

`w_AIC`
: Numeric. Weight for AIC evaluator. All weights must sum 1 (default: 0).

`w_c.index`
: Numeric. Weight for C-Index evaluator. All weights must sum 1 (default: 0).

`w_AUC`
: Numeric. Weight for AUC evaluator. All weights must sum 1 (default: 1).

`w_BRIER`
: Numeric. Weight for BRIER SCORE evaluator. All weights must sum 1 (default: 0).

`times`
: Numeric vector. Time points where the AUC will be evaluated. If NULL, a maximum of 'max_time_points' points will be selected equally distributed (default: NULL).

`max_time_points`
: Numeric. Maximum number of time points to use for evaluating the model (default: 15).

`MIN_AUC_INCREASE`
: Numeric. Minimum improvement between different cross validation models to continue evaluating higher values in the multiple tested parameters. If it is not reached for next 'MIN_COMP_TO_CHECK' models and the minimum 'MIN_AUC' value is reached, the evaluation stops (default: 0.01).

`MIN_AUC`
: Numeric. Minimum AUC desire to reach cross-validation models. If the minimum is reached, the evaluation could stop if the improvement does not reach an AUC higher than adding the 'MIN_AUC_INCREASE' value (default: 0.8).

`MIN_COMP_TO_CHECK`
: Numeric. Number of penalties/components to evaluate to check if the AUC improves. If for the next 'MIN_COMP_TO_CHECK' the AUC is not better and the 'MIN_AUC' is meet, the evaluation could stop (default: 3).

`pred.attr`
: Character. Way to evaluate the metric selected. Must be one of the following: "mean" or "median" (default: "mean").

pred.method	Character. AUC evaluation algorithm method for evaluate the model performance. Must be one of the following: "risksetROC", "survivalROC", "cenROC", "nsROC", "smoothROCtime_C", "smoothROCtime_I" (default: "cenROC").
max.iter	Numeric. Maximum number of iterations for PLS convergence (default: 200).
fast_mode	Logical. If fast_mode = TRUE, for each run, only one fold is evaluated simultaneously. If fast_mode = FALSE, for each run, all linear predictors are computed for test observations. Once all have their linear predictors, the evaluation is perform across all the observations together (default: FALSE).
MIN_EPV	Numeric. Minimum number of Events Per Variable (EPV) you want reach for the final cox model. Used to restrict the number of variables/components can be computed in final cox models. If the minimum is not meet, the model cannot be computed (default: 5).
return_models	Logical. Return all models computed in cross validation (default: FALSE).
returnData	Logical. Return original and normalized X and Y matrices (default: TRUE).
PARALLEL	Logical. Run the cross validation with multicore option. As many cores as your total cores - 1 will be used. It could lead to higher RAM consumption (default: FALSE).
verbose	Logical. If verbose = TRUE, extra messages could be displayed (default: FALSE).
seed	Number. Seed value for performing runs/folds divisions (default: 123).

Details

The function operates by partitioning the data into multiple subsets (folds) and iteratively holding out one subset for validation while training on the remaining subsets. The cross-validation process is repeated for a specified number of runs, ensuring a robust assessment of the model's performance. The function offers flexibility in terms of the number of PLS components, the range of variables considered, and the evaluation metrics used.

The function provides an option to center and scale the explanatory variables, which can be crucial for ensuring consistent performance, especially when the variables are measured on different scales. Additionally, the function incorporates features to handle near-zero and zero variance variables, which can be problematic in high-dimensional datasets.

For model evaluation, users can choose between various metrics, including AUC, c-index, and Brier Score. The function also allows for the specification of weights for these metrics, enabling users to prioritize certain metrics over others based on the research context.

The function's design also emphasizes computational efficiency. It offers a parallel processing option to expedite the cross-validation process, especially beneficial for large datasets. However, users should be cautious about potential high RAM consumption when using this option.

Value

Instance of class "Coxmos" and model "cv.MB.sPLS-DRCOX". `best_model_info`: A data.frame with the information for the best model. `df_results_folds`: A data.frame with fold-level information. `df_results_runs`: A data.frame with run-level information. `df_results_comps`: A data.frame with component-level information (for cv.coxEN, EN.alpha information).

lst_models: If return_models = TRUE, return a the list of all cross-validated models. pred.method: AUC evaluation algorithm method for evaluate the model performance.

opt.comp: Optimal component selected by the best_model. opt.nvar: Optimal number of variables selected by the best_model.

plot_AIC: AIC plot by each hyper-parameter. plot_c_index: C-Index plot by each hyper-parameter. plot_BRIER: Brier Score plot by each hyper-parameter. plot_AUC: AUC plot by each hyper-parameter.

class: Cross-Validated model class.

lst_train_indexes: List (of lists) of indexes for the observations used in each run/fold for train the models. lst_test_indexes: List (of lists) of indexes for the observations used in each run/fold for test the models.

time: time consumed for running the cross-validated function.

Author(s)

Pedro Salguero Garcia. Maintainer: pedsalga@upv.edu.es

Examples

```
data("X_multiomic")
data("Y_multiomic")
set.seed(123)
index_train <- caret::createDataPartition(Y_multiomic$event, p = .5, list = FALSE, times = 1)
X_train <- X_multiomic
X_train$mirna <- X_train$mirna[index_train,1:50]
X_train$proteomic <- X_train$proteomic[index_train,1:50]
Y_train <- Y_multiomic[index_train,]
cv.mb.splsdrcox_model <- cv.mb.splsdrcox(X_train, Y_train, max.ncomp = 2, vector = NULL,
n_run = 1, k_folds = 2, x.center = TRUE, x.scale = TRUE)
```

---

cv.sb.splsdrcox *SB.sPLS-DRCOX Cross-Validation*

Description

This function performs cross-validated sparse partial least squares single block for splsdrcox. The function returns the optimal number of components and the optimal sparsity penalty value based on cross-validation. The performance could be based on multiple metrics as Area Under the Curve (AUC), Brier Score or C-Index. Furthermore, the user could establish more than one metric simultaneously.

Usage

```
cv.sb.splsdrcox(
  X,
  Y,
  max.ncomp = 8,
  penalty.list = seq(0.1, 0.9, 0.2),
  n_run = 3,
  k_folds = 10,
  x.center = TRUE,
  x.scale = FALSE,
  remove_near_zero_variance = TRUE,
  remove_zero_variance = TRUE,
  toKeep.zv = NULL,
  remove_variance_at_fold_level = FALSE,
  remove_non_significant_models = FALSE,
  remove_non_significant = FALSE,
  alpha = 0.05,
  w_AIC = 0,
  w_c.index = 0,
  w_AUC = 1,
  w_BRIER = 0,
  times = NULL,
  max_time_points = 15,
  MIN_AUC_INCREASE = 0.01,
  MIN_AUC = 0.8,
  MIN_COMP_TO_CHECK = 3,
  pred.attr = "mean",
  pred.method = "cenROC",
  fast_mode = FALSE,
  MIN_EPV = 5,
  return_models = FALSE,
  returnData = FALSE,
  PARALLEL = FALSE,
  verbose = FALSE,
  seed = 123
)
```

Arguments

X	Numeric matrix or data.frame. Explanatory variables. Qualitative variables must be transform into binary variables.
Y	Numeric matrix or data.frame. Response variables. Object must have two columns named as "time" and "event". For event column, accepted values are: 0/1 or FALSE/TRUE for censored and event observations.
max.ncomp	Numeric. Maximum number of PLS components to compute for the cross validation (default: 8).
penalty.list	Numeric vector. Vector of penalty values. Penalty for sPLS-DRCOX. If penalty = 0 no penalty is applied, when penalty = 1 maximum penalty (no variables are

selected) based on 'plsRcox' penalty. Equal or greater than 1 cannot be selected (default: seq(0.1,0.9,0.2)).

`n_run` Numeric. Number of runs for cross validation (default: 3).

`k_folds` Numeric. Number of folds for cross validation (default: 10).

`x.center` Logical. If x.center = TRUE, X matrix is centered to zero means (default: TRUE).

`x.scale` Logical. If x.scale = TRUE, X matrix is scaled to unit variances (default: FALSE).

`remove_near_zero_variance`
Logical. If remove_near_zero_variance = TRUE, near zero variance variables will be removed (default: TRUE).

`remove_zero_variance`
Logical. If remove_zero_variance = TRUE, zero variance variables will be removed (default: TRUE).

`toKeep.zv` Character vector. Name of variables in X to not be deleted by (near) zero variance filtering (default: NULL).

`remove_variance_at_fold_level`
Logical. If remove_variance_at_fold_level = TRUE, (near) zero variance will be removed at fold level (default: FALSE).

`remove_non_significant_models`
Logical. If remove_non_significant_models = TRUE, non-significant models are removed before computing the evaluation. A non-significant model is a model with at least one component/variable with a P-Value higher than the alpha cutoff.

`remove_non_significant`
Logical. If remove_non_significant = TRUE, non-significant variables/components in final cox model will be removed until all variables are significant by forward selection (default: FALSE).

`alpha` Numeric. Numerical values are regarded as significant if they fall below the threshold (default: 0.05).

`w_AIC` Numeric. Weight for AIC evaluator. All weights must sum 1 (default: 0).

`w_c.index` Numeric. Weight for C-Index evaluator. All weights must sum 1 (default: 0).

`w_AUC` Numeric. Weight for AUC evaluator. All weights must sum 1 (default: 1).

`w_BRIER` Numeric. Weight for BRIER SCORE evaluator. All weights must sum 1 (default: 0).

`times` Numeric vector. Time points where the AUC will be evaluated. If NULL, a maximum of 'max_time_points' points will be selected equally distributed (default: NULL).

`max_time_points`
Numeric. Maximum number of time points to use for evaluating the model (default: 15).

`MIN_AUC_INCREASE`
Numeric. Minimum improvement between different cross validation models to continue evaluating higher values in the multiple tested parameters. If it is not reached for next 'MIN_COMP_TO_CHECK' models and the minimum 'MIN_AUC' value is reached, the evaluation stops (default: 0.01).

MIN_AUC	Numeric. Minimum AUC desire to reach cross-validation models. If the minimum is reached, the evaluation could stop if the improvement does not reach an AUC higher than adding the 'MIN_AUC_INCREASE' value (default: 0.8).
MIN_COMP_TO_CHECK	Numeric. Number of penalties/components to evaluate to check if the AUC improves. If for the next 'MIN_COMP_TO_CHECK' the AUC is not better and the 'MIN_AUC' is meet, the evaluation could stop (default: 3).
pred.attr	Character. Way to evaluate the metric selected. Must be one of the following: "mean" or "median" (default: "mean").
pred.method	Character. AUC evaluation algorithm method for evaluate the model performance. Must be one of the following: "risksetROC", "survivalROC", "cenROC", "nsROC", "smoothROCtime_C", "smoothROCtime_I" (default: "cenROC").
fast_mode	Logical. If fast_mode = TRUE, for each run, only one fold is evaluated simultaneously. If fast_mode = FALSE, for each run, all linear predictors are computed for test observations. Once all have their linear predictors, the evaluation is perform across all the observations together (default: FALSE).
MIN_EPV	Numeric. Minimum number of Events Per Variable (EPV) you want reach for the final cox model. Used to restrict the number of variables/components can be computed in final cox models. If the minimum is not meet, the model cannot be computed (default: 5).
return_models	Logical. Return all models computed in cross validation (default: FALSE).
returnData	Logical. Return original and normalized X and Y matrices (default: TRUE).
PARALLEL	Logical. Run the cross validation with multicore option. As many cores as your total cores - 1 will be used. It could lead to higher RAM consumption (default: FALSE).
verbose	Logical. If verbose = TRUE, extra messages could be displayed (default: FALSE).
seed	Number. Seed value for performing runs/folds divisions (default: 123).

Details

The cv.sb.splsdrcox function performs cross-validation for the single-block sparse partial least squares deviance residual Cox analysis. Cross-validation is a robust method to evaluate the performance of a statistical model by partitioning the original sample into a training set to train the model, and a test set to evaluate it. This helps in selecting the optimal hyperparameters for the model, such as the number of latent components (max.ncomp) and the penalty for variable selection (penalty.list).

The function systematically evaluates different combinations of hyperparameters by performing multiple runs and folds. For each combination, the dataset is divided into training and test sets based on the specified number of folds (k_folds). The model is then trained on the training set and evaluated on the test set. This process is repeated for the specified number of runs (n_run), ensuring a comprehensive evaluation of the model's performance across different partitions of the data.

Various evaluation metrics, such as AIC, C-Index, Brier Score, and AUC, are computed for each combination of hyperparameters. These metrics provide insights into the model's accuracy, discriminative ability, and calibration. The function then identifies the optimal hyperparameters that yield the best performance based on the specified evaluation metrics.

The function also offers flexibility in data preprocessing, such as centering and scaling of the explanatory variables, removal of near-zero variance variables, and more. Additionally, users can specify the AUC evaluation algorithm method (`pred.method`) and control the verbosity of the output (`verbose`).

The output provides a comprehensive overview of the cross-validation results, including detailed information at the fold, run, and component levels. Visualization tools, such as plots for AIC, C-Index, Brier Score, and AUC, are also provided to aid in understanding the model's performance across different hyperparameters.

In summary, the `cv.sb.splsdrcox` function offers a robust approach for hyperparameter tuning and model evaluation for the single-block sparse partial least squares deviance residual Cox analysis. It ensures that the final model is both accurate and generalizable to new data.

Value

Instance of class "Coxmos" and model "cv.SB.sPLS-DRCOX". `best_model_info`: A data.frame with the information for the best model. `df_results_folds`: A data.frame with fold-level information. `df_results_runs`: A data.frame with run-level information. `df_results_comps`: A data.frame with component-level information (for cv.coxEN, EN.alpha information).

`lst_models`: If return_models = TRUE, return a the list of all cross-validated models. `pred.method`: AUC evaluation algorithm method for evaluate the model performance.

`opt.comp`: Optimal component selected by the best_model. `opt.penalty`: Optimal penalty/penalty selected by the best_model. `opt.nvar`: Optimal number of variables selected by the best_model.

`plot_AIC`: AIC plot by each hyper-parameter. `plot_c_index`: C-Index plot by each hyper-parameter. `plot_BRIER`: Brier Score plot by each hyper-parameter. `plot_AUC`: AUC plot by each hyper-parameter.

`class`: Cross-Validated model class.

`lst_train_indexes`: List (of lists) of indexes for the observations used in each run/fold for train the models. `lst_test_indexes`: List (of lists) of indexes for the observations used in each run/fold for test the models.

`time`: time consumed for running the cross-validated function.

Author(s)

Pedro Salguero Garcia. Maintainer: pedsalga@upv.edu.es

Examples

```
data("X_multiomic")
data("Y_multiomic")
set.seed(123)
index_train <- caret::createDataPartition(Y_multiomic$event, p = .5, list = FALSE, times = 1)
X_train <- X_multiomic
X_train$mirna <- X_train$mirna[index_train,1:50]
X_train$proteomic <- X_train$proteomic[index_train,1:50]
Y_train <- Y_multiomic[index_train,]
cv.sb.splsdrcox_model <- cv.sb.splsdrcox(X_train, Y_train, max.ncomp = 2, penalty.list = c(0.5),
n_run = 1, k_folds = 2, x.center = TRUE, x.scale = TRUE)
```

`cv.sb.splsicox` *Cross validation cv.sb.splsicox*

Description

This function performs cross-validated sparse partial least squares single block for splsicox. The function returns the optimal number of components and the optimal sparsity penalty value based on cross-validation. The performance could be based on multiple metrics as Area Under the Curve (AUC), Brier Score or C-Index. Furthermore, the user could establish more than one metric simultaneously.

Usage

```
cv.sb.splsicox(
  X,
  Y,
  max.ncomp = 8,
  penalty.list = seq(0.1, 0.9, 0.2),
  n_run = 3,
  k_folds = 10,
  x.center = TRUE,
  x.scale = FALSE,
  remove_near_zero_variance = TRUE,
  remove_zero_variance = TRUE,
  toKeep.zv = NULL,
  remove_variance_at_fold_level = FALSE,
  remove_non_significant_models = FALSE,
  remove_non_significant = FALSE,
  alpha = 0.05,
  w_AIC = 0,
  w_c.index = 0,
  w_AUC = 1,
  w_BRIER = 0,
  times = NULL,
  max_time_points = 15,
  MIN_AUC_INCREASE = 0.01,
  MIN_AUC = 0.8,
  MIN_COMP_TO_CHECK = 3,
  pred.attr = "mean",
  pred.method = "cenROC",
  fast_mode = FALSE,
  MIN_EPV = 5,
  return_models = FALSE,
  returnData = FALSE,
  PARALLEL = FALSE,
  verbose = FALSE,
  seed = 123
```

)

Arguments

X	Numeric matrix or data.frame. Explanatory variables. Qualitative variables must be transform into binary variables.
Y	Numeric matrix or data.frame. Response variables. Object must have two columns named as "time" and "event". For event column, accepted values are: 0/1 or FALSE/TRUE for censored and event observations.
max.ncomp	Numeric. Maximum number of PLS components to compute for the cross validation (default: 8).
penalty.list	Numeric vector. Penalty for variable selection for the individual cox models. Variables with a lower P-Value than 1 - "penalty" in the individual cox analysis will be keep for the sPLS-ICOX approach (default: seq(0.1,0.9,0.2)).
n_run	Numeric. Number of runs for cross validation (default: 3).
k_folds	Numeric. Number of folds for cross validation (default: 10).
x.center	Logical. If x.center = TRUE, X matrix is centered to zero means (default: TRUE).
x.scale	Logical. If x.scale = TRUE, X matrix is scaled to unit variances (default: FALSE).
remove_near_zero_variance	Logical. If remove_near_zero_variance = TRUE, near zero variance variables will be removed (default: TRUE).
remove_zero_variance	Logical. If remove_zero_variance = TRUE, zero variance variables will be removed (default: TRUE).
toKeep.zv	Character vector. Name of variables in X to not be deleted by (near) zero variance filtering (default: NULL).
remove_variance_at_fold_level	Logical. If remove_variance_at_fold_level = TRUE, (near) zero variance will be removed at fold level (default: FALSE).
remove_non_significant_models	Logical. If remove_non_significant_models = TRUE, non-significant models are removed before computing the evaluation.
remove_non_significant	Logical. If remove_non_significant = TRUE, non-significant variables/components in final cox model will be removed until all variables are significant by forward selection (default: FALSE).
alpha	Numeric. Numerical values are regarded as significant if they fall below the threshold (default: 0.05).
w_AIC	Numeric. Weight for AIC evaluator. All weights must sum 1 (default: 0).
w_c.index	Numeric. Weight for C-Index evaluator. All weights must sum 1 (default: 0).
w_AUC	Numeric. Weight for AUC evaluator. All weights must sum 1 (default: 1).
w_BRIER	Numeric. Weight for BRIER SCORE evaluator. All weights must sum 1 (default: 0).

`times` Numeric vector. Time points where the AUC will be evaluated. If NULL, a maximum of 'max_time_points' points will be selected equally distributed (default: NULL).

`max_time_points` Numeric. Maximum number of time points to use for evaluating the model (default: 15).

`MIN_AUC_INCREASE` Numeric. Minimum improvement between different cross validation models to continue evaluating higher values in the multiple tested parameters. If it is not reached for next 'MIN_COMP_TO_CHECK' models and the minimum 'MIN_AUC' value is reached, the evaluation stops (default: 0.01).

`MIN_AUC` Numeric. Minimum AUC desire to reach cross-validation models. If the minimum is reached, the evaluation could stop if the improvement does not reach an AUC higher than adding the 'MIN_AUC_INCREASE' value (default: 0.8).

`MIN_COMP_TO_CHECK` Numeric. Number of penalties/components to evaluate to check if the AUC improves. If for the next 'MIN_COMP_TO_CHECK' the AUC is not better and the 'MIN_AUC' is meet, the evaluation could stop (default: 3).

`pred.attr` Character. Way to evaluate the metric selected. Must be one of the following: "mean" or "median" (default: "mean").

`pred.method` Character. AUC evaluation algorithm method for evaluate the model performance. Must be one of the following: "risksetROC", "survivalROC", "cenROC", "nsROC", "smoothROCtime_C", "smoothROCtime_I" (default: "cenROC").

`fast_mode` Logical. If fast_mode = TRUE, for each run, only one fold is evaluated simultaneously. If fast_mode = FALSE, for each run, all linear predictors are computed for test observations. Once all have their linear predictors, the evaluation is perform across all the observations together (default: FALSE).

`MIN_EPV` Numeric. Minimum number of Events Per Variable (EPV) you want reach for the final cox model. Used to restrict the number of variables/components can be computed in final cox models. If the minimum is not meet, the model cannot be computed (default: 5).

`return_models` Logical. Return all models computed in cross validation (default: FALSE).

`returnData` Logical. Return original and normalized X and Y matrices (default: TRUE).

`PARALLEL` Logical. Run the cross validation with multicore option. As many cores as your total cores - 1 will be used. It could lead to higher RAM consumption (default: FALSE).

`verbose` Logical. If verbose = TRUE, extra messages could be displayed (default: FALSE).

`seed` Number. Seed value for performing runs/folds divisions (default: 123).

Details

The `cv.sb.splsicox` function performs cross-validation for the single-block sparse partial least squares individual Cox analysis. While the function can handle datasets with multiple blocks, it processes each block individually, ensuring a detailed examination of each block's contribution

to the survival outcome. This is distinct from multiblock methods where all blocks are analyzed simultaneously.

In the context of this function, "single-block" means that each block of data is analyzed separately, one at a time. This approach is beneficial when different blocks represent distinct types or sources of data, allowing for a granular understanding of each block's significance without the interference of other blocks.

The cross-validation process involves partitioning the dataset into multiple subsets (folds) and then iteratively training the model on a subset of the data while validating it on the remaining data. This helps in determining the optimal hyperparameters for the model, such as the number of latent components and the penalty for variable selection.

The function offers flexibility in specifying various hyperparameters and options for data preprocessing. The output provides a comprehensive overview of the cross-validation results, including metrics like AIC, C-Index, Brier Score, and AUC for each hyper-parameter combination. Visualization tools are also provided to aid in understanding the model's performance across different hyperparameters.

In summary, the `cv.sb.splsicox` function offers a robust approach for determining the optimal parameters for the single-block sparse partial least squares individual Cox analysis, ensuring optimal feature selection, dimensionality reduction, and predictive modeling for each individual block in the dataset.

Value

Instance of class "Coxmos" and model "cv.SB.sPLS-ICOX". `best_model_info`: A data.frame with the information for the best model. `df_results_folds`: A data.frame with fold-level information. `df_results_runs`: A data.frame with run-level information. `df_results_comps`: A data.frame with component-level information (for cv.coxEN, EN.alpha information).

`lst_models`: If return_models = TRUE, return a the list of all cross-validated models. `pred.method`: AUC evaluation algorithm method for evaluate the model performance.

`opt.comp`: Optimal component selected by the best_model. `opt.penalty`: Optimal penalty value selected by the best_model.

`plot_AIC`: AIC plot by each hyper-parameter. `plot_c_index`: C-Index plot by each hyper-parameter. `plot_BRIER`: Brier Score plot by each hyper-parameter. `plot_AUC`: AUC plot by each hyper-parameter.

`class`: Cross-Validated model class.

`lst_train_indexes`: List (of lists) of indexes for the observations used in each run/fold for train the models. `lst_test_indexes`: List (of lists) of indexes for the observations used in each run/fold for test the models.

`time`: time consumed for running the cross-validated function.

Author(s)

Pedro Salguero Garcia. Maintainer: pedsalga@upv.edu.es

Examples

```
data("X_multiomic")
data("Y_multiomic")
set.seed(123)
index_train <- caret::createDataPartition(Y_multiomic$event, p = .5, list = FALSE, times = 1)
X_train <- X_multiomic
X_train$mirna <- X_train$mirna[index_train,1:50]
X_train$proteomic <- X_train$proteomic[index_train,1:50]
Y_train <- Y_multiomic[index_train,]
cv.sb.splsicox_model <- cv.sb.splsicox(X_train, Y_train, max.ncomp = 2, penalty.list = c(0.5),
n_run = 1, k_folds = 2, x.center = TRUE, x.scale = TRUE)
```

---

cv.splsdacox_dynamic *Cross validation splsdacox_dynamic*

Description

The cv.splsdacox_dynamic function performs cross-validation for the sPLS-DA-COX-Dynamic model. This model is designed to handle survival data, where the response variables are time-to-event and event/censoring indicators. The function offers a comprehensive set of parameters to fine-tune the cross-validation process, including options for data preprocessing, model evaluation, and parallel processing.

Usage

```
cv.splsdacox_dynamic(
  X,
  Y,
  max.ncomp = 8,
  vector = NULL,
  n_run = 3,
  k_folds = 10,
  x.center = TRUE,
  x.scale = FALSE,
  remove_near_zero_variance = TRUE,
  remove_zero_variance = TRUE,
  toKeep.zv = NULL,
  remove_variance_at_fold_level = FALSE,
  remove_non_significant_models = FALSE,
  remove_non_significant = FALSE,
  alpha = 0.05,
  MIN_NVAR = 10,
  MAX_NVAR = 1000,
  n.cut_points = 5,
  MIN_AUC_INCREASE = 0.01,
  EVAL_METHOD = "AUC",
  w_AIC = 0,
```

```
  w_c.index = 0,
  w_AUC = 1,
  w_BRIER = 0,
  times = NULL,
  max_time_points = 15,
  MIN_AUC = 0.8,
  MIN_COMP_TO_CHECK = 3,
  pred.attr = "mean",
  pred.method = "cenROC",
  fast_mode = FALSE,
  max.iter = 200,
  MIN_EPV = 5,
  return_models = FALSE,
  returnData = FALSE,
  PARALLEL = FALSE,
  verbose = FALSE,
  seed = 123
)
```

Arguments

X	Numeric matrix or data.frame. Explanatory variables. Qualitative variables must be transform into binary variables.
Y	Numeric matrix or data.frame. Response variables. Object must have two columns named as "time" and "event". For event column, accepted values are: 0/1 or FALSE/TRUE for censored and event observations.
max.ncomp	Numeric. Maximum number of PLS components to compute for the cross validation (default: 8).
vector	Numeric vector. Used for computing best number of variables. As many values as components have to be provided. If vector = NULL, an automatic detection is perform (default: NULL).
n_run	Numeric. Number of runs for cross validation (default: 3).
k_folds	Numeric. Number of folds for cross validation (default: 10).
x.center	Logical. If x.center = TRUE, X matrix is centered to zero means (default: TRUE).
x.scale	Logical. If x.scale = TRUE, X matrix is scaled to unit variances (default: FALSE).
remove_near_zero_variance	Logical. If remove_near_zero_variance = TRUE, near zero variance variables will be removed (default: TRUE).
remove_zero_variance	Logical. If remove_zero_variance = TRUE, zero variance variables will be removed (default: TRUE).
toKeep.zv	Character vector. Name of variables in X to not be deleted by (near) zero variance filtering (default: NULL).
remove_variance_at_fold_level	Logical. If remove_variance_at_fold_level = TRUE, (near) zero variance will be removed at fold level (default: FALSE).

remove_non_significant_models
: Logical. If remove_non_significant_models = TRUE, non-significant models are removed before computing the evaluation.

remove_non_significant
: Logical. If remove_non_significant = TRUE, non-significant variables/components in final cox model will be removed until all variables are significant by forward selection (default: FALSE).

alpha
: Numeric. Numerical values are regarded as significant if they fall below the threshold (default: 0.05).

MIN_NVAR
: Numeric. Minimum range size for computing cut points to select the best number of variables to use (default: 10).

MAX_NVAR
: Numeric. Maximum range size for computing cut points to select the best number of variables to use (default: 1000).

n.cut_points
: Numeric. Number of cut points for searching the optimal number of variables. If only two cut points are selected, minimum and maximum size are used. For MB approaches as many as n.cut_points^n.blocks models will be computed as minimum (default: 5).

MIN_AUC_INCREASE
: Numeric. Minimum improvement between different cross validation models to continue evaluating higher values in the multiple tested parameters. If it is not reached for next 'MIN_COMP_TO_CHECK' models and the minimum 'MIN_AUC' value is reached, the evaluation stops (default: 0.01).

EVAL_METHOD
: Character. If EVAL_METHOD = "AUC", AUC metric will be use to compute the best number of variables. In other case, c-index metric will be used (default: "AUC").

w_AIC
: Numeric. Weight for AIC evaluator. All weights must sum 1 (default: 0).

w_c.index
: Numeric. Weight for C-Index evaluator. All weights must sum 1 (default: 0).

w_AUC
: Numeric. Weight for AUC evaluator. All weights must sum 1 (default: 1).

w_BRIER
: Numeric. Weight for BRIER SCORE evaluator. All weights must sum 1 (default: 0).

times
: Numeric vector. Time points where the AUC will be evaluated. If NULL, a maximum of 'max_time_points' points will be selected equally distributed (default: NULL).

max_time_points
: Numeric. Maximum number of time points to use for evaluating the model (default: 15).

MIN_AUC
: Numeric. Minimum AUC desire to reach cross-validation models. If the minimum is reached, the evaluation could stop if the improvement does not reach an AUC higher than adding the 'MIN_AUC_INCREASE' value (default: 0.8).

MIN_COMP_TO_CHECK
: Numeric. Number of penalties/components to evaluate to check if the AUC improves. If for the next 'MIN_COMP_TO_CHECK' the AUC is not better and the 'MIN_AUC' is meet, the evaluation could stop (default: 3).

pred.attr
: Character. Way to evaluate the metric selected. Must be one of the following: "mean" or "median" (default: "mean").

pred.method	Character. AUC evaluation algorithm method for evaluate the model performance. Must be one of the following: "risksetROC", "survivalROC", "cenROC", "nsROC", "smoothROCtime_C", "smoothROCtime_I" (default: "cenROC").
fast_mode	Logical. If fast_mode = TRUE, for each run, only one fold is evaluated simultaneously. If fast_mode = FALSE, for each run, all linear predictors are computed for test observations. Once all have their linear predictors, the evaluation is perform across all the observations together (default: FALSE).
max.iter	Numeric. Maximum number of iterations for PLS convergence (default: 200).
MIN_EPV	Numeric. Minimum number of Events Per Variable (EPV) you want reach for the final cox model. Used to restrict the number of variables/components can be computed in final cox models. If the minimum is not meet, the model cannot be computed (default: 5).
return_models	Logical. Return all models computed in cross validation (default: FALSE).
returnData	Logical. Return original and normalized X and Y matrices (default: TRUE).
PARALLEL	Logical. Run the cross validation with multicore option. As many cores as your total cores - 1 will be used. It could lead to higher RAM consumption (default: FALSE).
verbose	Logical. If verbose = TRUE, extra messages could be displayed (default: FALSE).
seed	Number. Seed value for performing runs/folds divisions (default: 123).

Details

The function begins by ensuring that the required libraries for evaluation metrics are installed. It then checks the validity of the input parameters, such as ensuring that the response variables have the appropriate column names ("time" and "event") and that the evaluation weights sum to 1.

Data preprocessing steps include the potential removal of variables with zero or near-zero variance, and the transformation of explanatory variables to ensure they are centered or scaled as specified. The function also provides an option to remove variables based on their coefficient of variation.

The core of the function revolves around the cross-validation process. Data is split into training and test sets for each run and fold. For each combination of run, fold, and specified number of PLS components, a sPLS-DA-COX-Dynamic model is trained. The performance of these models is then evaluated using a combination of metrics, including the Akaike Information Criterion (AIC), C-index, Brier Score, and Area Under the Curve (AUC). The function provides flexibility in choosing the evaluation metric and its method.

Value

Instance of class "Coxmos" and model "cv.sPLS-DACOX-Dynamic". `best_model_info`: A data.frame with the information for the best model. `df_results_folds`: A data.frame with fold-level information. `df_results_runs`: A data.frame with run-level information. `df_results_comps`: A data.frame with component-level information (for cv.coxEN, EN.alpha information).

`lst_models`: If return_models = TRUE, return a the list of all cross-validated models. `pred.method`: AUC evaluation algorithm method for evaluate the model performance.

`opt.comp`: Optimal component selected by the best_model. `opt.nvar`: Optimal number of variables selected by the best_model.

`plot_AIC`: AIC plot by each hyper-parameter. `plot_c_index`: C-Index plot by each hyper-parameter. `plot_BRIER`: Brier Score plot by each hyper-parameter. `plot_AUC`: AUC plot by each hyper-parameter.

`class`: Cross-Validated model class.

`lst_train_indexes`: List (of lists) of indexes for the observations used in each run/fold for train the models. `lst_test_indexes`: List (of lists) of indexes for the observations used in each run/fold for test the models.

`time`: time consumed for running the cross-validated function.

Author(s)

Pedro Salguero Garcia. Maintainer: pedsalga@upv.edu.es

Examples

```
data("X_proteomic")
data("Y_proteomic")
set.seed(123)
index_train <- caret::createDataPartition(Y_proteomic$event, p = .5, list = FALSE, times = 1)
X_train <- X_proteomic[index_train,1:50]
Y_train <- Y_proteomic[index_train,]
cv.splsdacox_dynamic_model <- cv.splsdacox_dynamic(X_train, Y_train, max.ncomp = 2, vector = NULL,
n_run = 1, k_folds = 2, x.center = TRUE, x.scale = TRUE)
```

---

cv.splsdrcox *sPLS-DRCOX Cross-Validation*

Description

This function performs cross-validated sparse partial least squares DRCox (sPLS-DRCOX). The function returns the optimal number of components and the optimal sparsity penalty value based on cross-validation. The performance could be based on multiple metrics as Area Under the Curve (AUC), Brier Score or C-Index. Furthermore, the user could establish more than one metric simultaneously.

Usage

```
cv.splsdrcox(
  X,
  Y,
  max.ncomp = 8,
  penalty.list = seq(0.1, 0.9, 0.2),
  n_run = 3,
  k_folds = 10,
  x.center = TRUE,
  x.scale = FALSE,
  remove_near_zero_variance = TRUE,
```

```
  remove_zero_variance = TRUE,
  toKeep.zv = NULL,
  remove_variance_at_fold_level = FALSE,
  remove_non_significant_models = FALSE,
  remove_non_significant = FALSE,
  alpha = 0.05,
  w_AIC = 0,
  w_c.index = 0,
  w_AUC = 1,
  w_BRIER = 0,
  times = NULL,
  max_time_points = 15,
  MIN_AUC_INCREASE = 0.01,
  MIN_AUC = 0.8,
  MIN_COMP_TO_CHECK = 3,
  pred.attr = "mean",
  pred.method = "cenROC",
  fast_mode = FALSE,
  MIN_EPV = 5,
  return_models = FALSE,
  returnData = FALSE,
  PARALLEL = FALSE,
  verbose = FALSE,
  seed = 123
)
```

Arguments

X	Numeric matrix or data.frame. Explanatory variables. Qualitative variables must be transform into binary variables.
Y	Numeric matrix or data.frame. Response variables. Object must have two columns named as "time" and "event". For event column, accepted values are: 0/1 or FALSE/TRUE for censored and event observations.
max.ncomp	Numeric. Maximum number of PLS components to compute for the cross validation (default: 8).
penalty.list	Numeric vector. Vector of penalty values. Penalty for sPLS-DRCOX. If penalty = 0 no penalty is applied, when penalty = 1 maximum penalty (no variables are selected) based on 'plsRcox' penalty. Equal or greater than 1 cannot be selected (default: seq(0.1,0.9,0.2)).
n_run	Numeric. Number of runs for cross validation (default: 3).
k_folds	Numeric. Number of folds for cross validation (default: 10).
x.center	Logical. If x.center = TRUE, X matrix is centered to zero means (default: TRUE).
x.scale	Logical. If x.scale = TRUE, X matrix is scaled to unit variances (default: FALSE).
remove_near_zero_variance	Logical. If remove_near_zero_variance = TRUE, near zero variance variables will be removed (default: TRUE).

remove_zero_variance
: Logical. If remove_zero_variance = TRUE, zero variance variables will be removed (default: TRUE).

toKeep.zv
: Character vector. Name of variables in X to not be deleted by (near) zero variance filtering (default: NULL).

remove_variance_at_fold_level
: Logical. If remove_variance_at_fold_level = TRUE, (near) zero variance will be removed at fold level (default: FALSE).

remove_non_significant_models
: Logical. If remove_non_significant_models = TRUE, non-significant models are removed before computing the evaluation. A non-significant model is a model with at least one component/variable with a P-Value higher than the alpha cutoff.

remove_non_significant
: Logical. If remove_non_significant = TRUE, non-significant variables/components in final cox model will be removed until all variables are significant by forward selection (default: FALSE).

alpha
: Numeric. Numerical values are regarded as significant if they fall below the threshold (default: 0.05).

w_AIC
: Numeric. Weight for AIC evaluator. All weights must sum 1 (default: 0).

w_c.index
: Numeric. Weight for C-Index evaluator. All weights must sum 1 (default: 0).

w_AUC
: Numeric. Weight for AUC evaluator. All weights must sum 1 (default: 1).

w_BRIER
: Numeric. Weight for BRIER SCORE evaluator. All weights must sum 1 (default: 0).

times
: Numeric vector. Time points where the AUC will be evaluated. If NULL, a maximum of 'max_time_points' points will be selected equally distributed (default: NULL).

max_time_points
: Numeric. Maximum number of time points to use for evaluating the model (default: 15).

MIN_AUC_INCREASE
: Numeric. Minimum improvement between different cross validation models to continue evaluating higher values in the multiple tested parameters. If it is not reached for next 'MIN_COMP_TO_CHECK' models and the minimum 'MIN_AUC' value is reached, the evaluation stops (default: 0.01).

MIN_AUC
: Numeric. Minimum AUC desire to reach cross-validation models. If the minimum is reached, the evaluation could stop if the improvement does not reach an AUC higher than adding the 'MIN_AUC_INCREASE' value (default: 0.8).

MIN_COMP_TO_CHECK
: Numeric. Number of penalties/components to evaluate to check if the AUC improves. If for the next 'MIN_COMP_TO_CHECK' the AUC is not better and the 'MIN_AUC' is meet, the evaluation could stop (default: 3).

pred.attr
: Character. Way to evaluate the metric selected. Must be one of the following: "mean" or "median" (default: "mean").

pred.method	Character. AUC evaluation algorithm method for evaluate the model performance. Must be one of the following: "risksetROC", "survivalROC", "cenROC", "nsROC", "smoothROCtime_C", "smoothROCtime_I" (default: "cenROC").
fast_mode	Logical. If fast_mode = TRUE, for each run, only one fold is evaluated simultaneously. If fast_mode = FALSE, for each run, all linear predictors are computed for test observations. Once all have their linear predictors, the evaluation is perform across all the observations together (default: FALSE).
MIN_EPV	Numeric. Minimum number of Events Per Variable (EPV) you want reach for the final cox model. Used to restrict the number of variables/components can be computed in final cox models. If the minimum is not meet, the model cannot be computed (default: 5).
return_models	Logical. Return all models computed in cross validation (default: FALSE).
returnData	Logical. Return original and normalized X and Y matrices (default: TRUE).
PARALLEL	Logical. Run the cross validation with multicore option. As many cores as your total cores - 1 will be used. It could lead to higher RAM consumption (default: FALSE).
verbose	Logical. If verbose = TRUE, extra messages could be displayed (default: FALSE).
seed	Number. Seed value for performing runs/folds divisions (default: 123).

Details

The sPLS-DRCOX Cross-Validation function offers a robust approach to fine-tune the hyperparameters of the sPLS-DRCOX model, ensuring optimal performance in survival analysis tasks. By systematically evaluating different combinations of hyperparameters, this function identifies the best model configuration that minimizes prediction error.

Cross-validation is a crucial step in survival analysis, especially when dealing with high-dimensional datasets. It provides an unbiased assessment of the model's generalization capability, safeguarding against overfitting. This function employs a k-fold cross-validation strategy, partitioning the data into multiple subsets (folds) and iteratively using each fold as a test set while the remaining folds serve as training data.

One of the primary strengths of this function is its flexibility. Users can specify a range of values for the number of PLS components and the penalty parameter penalty. The function then evaluates all possible combinations, returning the optimal configuration that yields the best predictive performance.

Additionally, the function offers advanced features like parallel processing for faster computation, and the ability to return all models from the cross-validation process. This is particularly useful for in-depth analysis and comparisons.

The output provides comprehensive insights, including performance metrics for each fold, run, and hyperparameter combination. Visualization plots like AIC, C-Index, Brier Score, and AUC plots further aid in understanding the model's performance across different configurations.

Value

Instance of class "Coxmos" and model "cv.sPLS-DRCOX". best_model_info: A data.frame with the information for the best model. df_results_folds: A data.frame with fold-level information.

df_results_runs: A data.frame with run-level information. df_results_comps: A data.frame with component-level information (for cv.coxEN, EN.alpha information).

lst_models: If return_models = TRUE, return a the list of all cross-validated models. pred.method: AUC evaluation algorithm method for evaluate the model performance.

opt.comp: Optimal component selected by the best_model. opt.penalty: Optimal penalty/penalty selected by the best_model. opt.nvar: Optimal number of variables selected by the best_model.

plot_AIC: AIC plot by each hyper-parameter. plot_c_index: C-Index plot by each hyper-parameter. plot_BRIER: Brier Score plot by each hyper-parameter. plot_AUC: AUC plot by each hyper-parameter.

class: Cross-Validated model class.

lst_train_indexes: List (of lists) of indexes for the observations used in each run/fold for train the models. lst_test_indexes: List (of lists) of indexes for the observations used in each run/fold for test the models.

time: time consumed for running the cross-validated function.

Author(s)

Pedro Salguero Garcia. Maintainer: pedsalga@upv.edu.es

Examples

```
data("X_proteomic")
data("Y_proteomic")
set.seed(123)
index_train <- caret::createDataPartition(Y_proteomic$event, p = .5, list = FALSE, times = 1)
X_train <- X_proteomic[index_train,1:50]
Y_train <- Y_proteomic[index_train,]
cv.splsdrcox_model <- cv.splsdrcox(X_train, Y_train, max.ncomp = 2, penalty.list = c(0.1),
n_run = 1, k_folds = 2, x.center = TRUE, x.scale = TRUE)
```

---

cv.splsdrcox_dynamic *Cross validation sPLS-DRCOX*

Description

The function cv.splsdrcox_dynamic conducts a cross-validation for the sPLS-DRCOX model, which is a specialized model tailored for survival analysis. The function aims to optimize the model's performance by determining the best number of PLS components and variables through cross-validation.

Usage

```
cv.splsdrcox_dynamic(
  X,
  Y,
  max.ncomp = 8,
```

```
  vector = NULL,
  n_run = 3,
  k_folds = 10,
  x.center = TRUE,
  x.scale = FALSE,
  remove_near_zero_variance = TRUE,
  remove_zero_variance = TRUE,
  toKeep.zv = NULL,
  remove_variance_at_fold_level = FALSE,
  remove_non_significant_models = FALSE,
  remove_non_significant = FALSE,
  alpha = 0.05,
  MIN_NVAR = 10,
  MAX_NVAR = 1000,
  n.cut_points = 5,
  MIN_AUC_INCREASE = 0.01,
  EVAL_METHOD = "AUC",
  w_AIC = 0,
  w_c.index = 0,
  w_AUC = 1,
  w_BRIER = 0,
  times = NULL,
  max_time_points = 15,
  MIN_AUC = 0.8,
  MIN_COMP_TO_CHECK = 3,
  pred.attr = "mean",
  pred.method = "cenROC",
  fast_mode = FALSE,
  max.iter = 200,
  MIN_EPV = 5,
  return_models = FALSE,
  returnData = FALSE,
  PARALLEL = FALSE,
  verbose = FALSE,
  seed = 123
)
```

Arguments

X	Numeric matrix or data.frame. Explanatory variables. Qualitative variables must be transform into binary variables.
Y	Numeric matrix or data.frame. Response variables. Object must have two columns named as "time" and "event". For event column, accepted values are: 0/1 or FALSE/TRUE for censored and event observations.
max.ncomp	Numeric. Maximum number of PLS components to compute for the cross validation (default: 8).
vector	Numeric vector. Used for computing best number of variables. As many values as components have to be provided. If vector = NULL, an automatic detection

is perform (default: NULL).

`n_run` Numeric. Number of runs for cross validation (default: 3).

`k_folds` Numeric. Number of folds for cross validation (default: 10).

`x.center` Logical. If x.center = TRUE, X matrix is centered to zero means (default: TRUE).

`x.scale` Logical. If x.scale = TRUE, X matrix is scaled to unit variances (default: FALSE).

`remove_near_zero_variance`
Logical. If remove_near_zero_variance = TRUE, near zero variance variables will be removed (default: TRUE).

`remove_zero_variance`
Logical. If remove_zero_variance = TRUE, zero variance variables will be removed (default: TRUE).

`toKeep.zv` Character vector. Name of variables in X to not be deleted by (near) zero variance filtering (default: NULL).

`remove_variance_at_fold_level`
Logical. If remove_variance_at_fold_level = TRUE, (near) zero variance will be removed at fold level (default: FALSE).

`remove_non_significant_models`
Logical. If remove_non_significant_models = TRUE, non-significant models are removed before computing the evaluation. A non-significant model is a model with at least one component/variable with a P-Value higher than the alpha cutoff.

`remove_non_significant`
Logical. If remove_non_significant = TRUE, non-significant variables/components in final cox model will be removed until all variables are significant by forward selection (default: FALSE).

`alpha` Numeric. Numerical values are regarded as significant if they fall below the threshold (default: 0.05).

`MIN_NVAR` Numeric. Minimum range size for computing cut points to select the best number of variables to use (default: 10).

`MAX_NVAR` Numeric. Maximum range size for computing cut points to select the best number of variables to use (default: 1000).

`n.cut_points` Numeric. Number of cut points for searching the optimal number of variables. If only two cut points are selected, minimum and maximum size are used. For MB approaches as many as n.cut_points^n.blocks models will be computed as minimum (default: 5).

`MIN_AUC_INCREASE`
Numeric. Minimum improvement between different cross validation models to continue evaluating higher values in the multiple tested parameters. If it is not reached for next 'MIN_COMP_TO_CHECK' models and the minimum 'MIN_AUC' value is reached, the evaluation stops (default: 0.01).

`EVAL_METHOD` Character. If EVAL_METHOD = "AUC", AUC metric will be use to compute the best number of variables. In other case, c-index metric will be used (default: "AUC").

w_AIC Numeric. Weight for AIC evaluator. All weights must sum 1 (default: 0).

w_c.index Numeric. Weight for C-Index evaluator. All weights must sum 1 (default: 0).

w_AUC Numeric. Weight for AUC evaluator. All weights must sum 1 (default: 1).

w_BRIER Numeric. Weight for BRIER SCORE evaluator. All weights must sum 1 (default: 0).

times Numeric vector. Time points where the AUC will be evaluated. If NULL, a maximum of 'max_time_points' points will be selected equally distributed (default: NULL).

max_time_points
Numeric. Maximum number of time points to use for evaluating the model (default: 15).

MIN_AUC Numeric. Minimum AUC desire to reach cross-validation models. If the minimum is reached, the evaluation could stop if the improvement does not reach an AUC higher than adding the 'MIN_AUC_INCREASE' value (default: 0.8).

MIN_COMP_TO_CHECK
Numeric. Number of penalties/components to evaluate to check if the AUC improves. If for the next 'MIN_COMP_TO_CHECK' the AUC is not better and the 'MIN_AUC' is meet, the evaluation could stop (default: 3).

pred.attr Character. Way to evaluate the metric selected. Must be one of the following: "mean" or "median" (default: "mean").

pred.method Character. AUC evaluation algorithm method for evaluate the model performance. Must be one of the following: "risksetROC", "survivalROC", "cenROC", "nsROC", "smoothROCtime_C", "smoothROCtime_I" (default: "cenROC").

fast_mode Logical. If fast_mode = TRUE, for each run, only one fold is evaluated simultaneously. If fast_mode = FALSE, for each run, all linear predictors are computed for test observations. Once all have their linear predictors, the evaluation is perform across all the observations together (default: FALSE).

max.iter Numeric. Maximum number of iterations for PLS convergence (default: 200).

MIN_EPV Numeric. Minimum number of Events Per Variable (EPV) you want reach for the final cox model. Used to restrict the number of variables/components can be computed in final cox models. If the minimum is not meet, the model cannot be computed (default: 5).

return_models Logical. Return all models computed in cross validation (default: FALSE).

returnData Logical. Return original and normalized X and Y matrices (default: TRUE).

PARALLEL Logical. Run the cross validation with multicore option. As many cores as your total cores - 1 will be used. It could lead to higher RAM consumption (default: FALSE).

verbose Logical. If verbose = TRUE, extra messages could be displayed (default: FALSE).

seed Number. Seed value for performing runs/folds divisions (default: 123).

Details

The cv.splsdrcox_dynamic function is designed to perform cross-validation for the sPLS-DRCOX model, a specialized model for survival analysis. The function's primary objective is to identify the optimal number of PLS components and variables that yield the best model performance.

The function accepts both numeric matrices and data frames for explanatory (X) and response (Y) variables. It is essential to ensure that qualitative variables in X are transformed into binary format. The response variable Y should have two columns: "time" and "event". The event column should contain binary values, where 0/1 or FALSE/TRUE represent censored and event observations, respectively.

The cross-validation process is controlled by several parameters, including the maximum number of PLS components (max.ncomp), the number of runs (n_run), and the number of folds (k_folds). The function also provides options for data preprocessing, such as centering and scaling of the X matrix, and removal of variables with near-zero or zero variance.

Significance testing is incorporated into the model evaluation process. Users can specify the alpha threshold (alpha) for determining significance. Non-significant models or variables can be optionally removed from the evaluation based on user-defined criteria.

The function also offers flexibility in model evaluation metrics. Users can choose between different metrics such as AUC, AIC, C-Index, and Brier Score. The importance of each metric in the evaluation can be controlled using weights (w_AIC, w_c.index, w_AUC, w_BRIER).

For computational efficiency, the function provides an option to run the cross-validation in parallel (PARALLEL). Additionally, verbose logging can be enabled to display extra messages during the execution.

Value

Instance of class "Coxmos" and model "cv.sPLS-DRCOX-Dynamic". `best_model_info`: A data.frame with the information for the best model. `df_results_folds`: A data.frame with fold-level information. `df_results_runs`: A data.frame with run-level information. `df_results_comps`: A data.frame with component-level information (for cv.coxEN, EN.alpha information).

`lst_models`: If return_models = TRUE, return a the list of all cross-validated models. `pred.method`: AUC evaluation algorithm method for evaluate the model performance.

`opt.comp`: Optimal component selected by the best_model. `opt.nvar`: Optimal number of variables selected by the best_model.

`plot_AIC`: AIC plot by each hyper-parameter. `plot_c_index`: C-Index plot by each hyper-parameter. `plot_BRIER`: Brier Score plot by each hyper-parameter. `plot_AUC`: AUC plot by each hyper-parameter.

`class`: Cross-Validated model class.

`lst_train_indexes`: List (of lists) of indexes for the observations used in each run/fold for train the models. `lst_test_indexes`: List (of lists) of indexes for the observations used in each run/fold for test the models.

`time`: time consumed for running the cross-validated function.

Author(s)

Pedro Salguero Garcia. Maintainer: pedsalga@upv.edu.es

Examples

```
data("X_proteomic")
data("Y_proteomic")
set.seed(123)
index_train <- caret::createDataPartition(Y_proteomic$event, p = .5, list = FALSE, times = 1)
X_train <- X_proteomic[index_train,1:20]
Y_train <- Y_proteomic[index_train,]
cv.splsdrcox_dynamic_model <- cv.splsdrcox_dynamic(X_train, Y_train, max.ncomp = 1, vector = NULL,
n_run = 1, k_folds = 2, x.center = TRUE, x.scale = TRUE)
```

`cv.splsicox` *sPLS-ICOX Cross-Validation*

Description

This function performs cross-validated sparse partial least squares Cox (sPLS-ICOX). The function returns the optimal number of components and the optimal sparsity penalty value based on cross-validation. The performance could be based on multiple metrics as Area Under the Curve (AUC), Brier Score or C-Index. Furthermore, the user could establish more than one metric simultaneously.

Usage

```
cv.splsicox(
  X,
  Y,
  max.ncomp = 8,
  penalty.list = seq(0, 0.9, 0.1),
  n_run = 3,
  k_folds = 10,
  x.center = TRUE,
  x.scale = FALSE,
  remove_near_zero_variance = TRUE,
  remove_zero_variance = TRUE,
  toKeep.zv = NULL,
  remove_variance_at_fold_level = FALSE,
  remove_non_significant_models = FALSE,
  remove_non_significant = FALSE,
  alpha = 0.05,
  w_AIC = 0,
  w_c.index = 0,
  w_AUC = 1,
  w_BRIER = 0,
  times = NULL,
  max_time_points = 15,
  MIN_AUC_INCREASE = 0.05,
  MIN_AUC = 0.8,
  MIN_COMP_TO_CHECK = 3,
```

```
  pred.attr = "mean",
  pred.method = "cenROC",
  fast_mode = FALSE,
  MIN_EPV = 5,
  return_models = FALSE,
  returnData = FALSE,
  PARALLEL = FALSE,
  verbose = FALSE,
  seed = 123
)
```

Arguments

X	Numeric matrix or data.frame. Explanatory variables. Qualitative variables must be transform into binary variables.
Y	Numeric matrix or data.frame. Response variables. Object must have two columns named as "time" and "event". For event column, accepted values are: 0/1 or FALSE/TRUE for censored and event observations.
max.ncomp	Numeric. Maximum number of PLS components to compute for the cross validation (default: 8).
penalty.list	Numeric vector. Penalty for variable selection for the individual cox models. Variables with a lower P-Value than 1 - "penalty" in the individual cox analysis will be keep for the sPLS-ICOX approach (default: seq(0.1,0.9,0.2)).
n_run	Numeric. Number of runs for cross validation (default: 3).
k_folds	Numeric. Number of folds for cross validation (default: 10).
x.center	Logical. If x.center = TRUE, X matrix is centered to zero means (default: TRUE).
x.scale	Logical. If x.scale = TRUE, X matrix is scaled to unit variances (default: FALSE).
remove_near_zero_variance	Logical. If remove_near_zero_variance = TRUE, near zero variance variables will be removed (default: TRUE).
remove_zero_variance	Logical. If remove_zero_variance = TRUE, zero variance variables will be removed (default: TRUE).
toKeep.zv	Character vector. Name of variables in X to not be deleted by (near) zero variance filtering (default: NULL).
remove_variance_at_fold_level	Logical. If remove_variance_at_fold_level = TRUE, (near) zero variance will be removed at fold level (default: FALSE).
remove_non_significant_models	Logical. If remove_non_significant_models = TRUE, non-significant models are removed before computing the evaluation. A non-significant model is a model with at least one component/variable with a P-Value higher than the alpha cutoff.

remove_non_significant
: Logical. If remove_non_significant = TRUE, non-significant variables/components in final cox model will be removed until all variables are significant by forward selection (default: FALSE).

alpha
: Numeric. Numerical values are regarded as significant if they fall below the threshold (default: 0.05).

w_AIC
: Numeric. Weight for AIC evaluator. All weights must sum 1 (default: 0).

w_c.index
: Numeric. Weight for C-Index evaluator. All weights must sum 1 (default: 0).

w_AUC
: Numeric. Weight for AUC evaluator. All weights must sum 1 (default: 1).

w_BRIER
: Numeric. Weight for BRIER SCORE evaluator. All weights must sum 1 (default: 0).

times
: Numeric vector. Time points where the AUC will be evaluated. If NULL, a maximum of 'max_time_points' points will be selected equally distributed (default: NULL).

max_time_points
: Numeric. Maximum number of time points to use for evaluating the model (default: 15).

MIN_AUC_INCREASE
: Numeric. Minimum improvement between different cross validation models to continue evaluating higher values in the multiple tested parameters. If it is not reached for next 'MIN_COMP_TO_CHECK' models and the minimum 'MIN_AUC' value is reached, the evaluation stops (default: 0.01).

MIN_AUC
: Numeric. Minimum AUC desire to reach cross-validation models. If the minimum is reached, the evaluation could stop if the improvement does not reach an AUC higher than adding the 'MIN_AUC_INCREASE' value (default: 0.8).

MIN_COMP_TO_CHECK
: Numeric. Number of penalties/components to evaluate to check if the AUC improves. If for the next 'MIN_COMP_TO_CHECK' the AUC is not better and the 'MIN_AUC' is meet, the evaluation could stop (default: 3).

pred.attr
: Character. Way to evaluate the metric selected. Must be one of the following: "mean" or "median" (default: "mean").

pred.method
: Character. AUC evaluation algorithm method for evaluate the model performance. Must be one of the following: "risksetROC", "survivalROC", "cenROC", "nsROC", "smoothROCtime_C", "smoothROCtime_I" (default: "cenROC").

fast_mode
: Logical. If fast_mode = TRUE, for each run, only one fold is evaluated simultaneously. If fast_mode = FALSE, for each run, all linear predictors are computed for test observations. Once all have their linear predictors, the evaluation is perform across all the observations together (default: FALSE).

MIN_EPV
: Numeric. Minimum number of Events Per Variable (EPV) you want reach for the final cox model. Used to restrict the number of variables/components can be computed in final cox models. If the minimum is not meet, the model cannot be computed (default: 5).

return_models
: Logical. Return all models computed in cross validation (default: FALSE).

returnData	Logical. Return original and normalized X and Y matrices (default: TRUE).
PARALLEL	Logical. Run the cross validation with multicore option. As many cores as your total cores - 1 will be used. It could lead to higher RAM consumption (default: FALSE).
verbose	Logical. If verbose = TRUE, extra messages could be displayed (default: FALSE).
seed	Number. Seed value for performing runs/folds divisions (default: 123).

Details

The sPLS-ICOX Cross-Validation function offers a systematic approach to determine the optimal hyperparameters for the sparse partial least squares Cox (sPLS-ICOX) model through cross-validation. This function aims to identify the best combination of the number of PLS components (max.ncomp) and the sparsity penalty (penalty.list) by evaluating model performance across multiple metrics such as Area Under the Curve (AUC), Brier Score, and C-Index.

Cross-validation is executed through a series of runs (n_run) and folds (k_folds), ensuring a robust assessment of model performance. The function provides flexibility in defining the evaluation criteria, allowing users to set weights for different metrics (w_AIC, w_c.index, w_AUC, w_BRIER) and to specify the desired evaluation method (pred.method).

An essential feature of this function is its ability to halt the evaluation process based on predefined conditions. If the improvement in AUC across successive models does not surpass the MIN_AUC_INCREASE threshold or if the desired AUC (MIN_AUC) is achieved, the evaluation can be terminated early, optimizing computational efficiency.

The function also incorporates various data preprocessing options, emphasizing the importance of data quality in model performance. For instance, near-zero and zero variance variables can be removed either globally or at the fold level. Additionally, the function can handle multicore processing (PARALLEL option) to expedite the cross-validation process.

Upon completion, the function returns a comprehensive output, including detailed information about the best model, performance metrics at various levels (fold, run, component), and optionally, all cross-validated models.

Value

Instance of class "Coxmos" and model "cv.sPLS-ICOX". best_model_info: A data.frame with the information for the best model. df_results_folds: A data.frame with fold-level information. df_results_runs: A data.frame with run-level information. df_results_comps: A data.frame with component-level information (for cv.coxEN, EN.alpha information).

lst_models: If return_models = TRUE, return a the list of all cross-validated models. pred.method: AUC evaluation algorithm method for evaluate the model performance.

opt.comp: Optimal component selected by the best_model. opt.penalty: Optimal penalty value selected by the best_model.

plot_AIC: AIC plot by each hyper-parameter. plot_c_index: C-Index plot by each hyper-parameter. plot_BRIER: Brier Score plot by each hyper-parameter. plot_AUC: AUC plot by each hyper-parameter.

class: Cross-Validated model class.

`lst_train_indexes`: List (of lists) of indexes for the observations used in each run/fold for train the models. `lst_test_indexes`: List (of lists) of indexes for the observations used in each run/fold for test the models.

`time`: time consumed for running the cross-validated function.

Author(s)

Pedro Salguero Garcia. Maintainer: pedsalga@upv.edu.es

Examples

```
data("X_proteomic")
data("Y_proteomic")
set.seed(123)
index_train <- caret::createDataPartition(Y_proteomic$event, p = .5, list = FALSE, times = 1)
X_train <- X_proteomic[index_train,1:50]
Y_train <- Y_proteomic[index_train,]
cv.splsicox_model <- cv.splsicox(X_train, Y_train, max.ncomp = 2, penalty.list = c(0.1),
n_run = 1, k_folds = 2, x.center = TRUE, x.scale = TRUE)
```

---

deleteNearZeroCoefficientOfVariation

deleteNearZeroCoefficientOfVariation

Description

Filters out variables from a dataset that exhibit a coefficient of variation below a specified threshold, ensuring the retention of variables with meaningful variability.

Usage

```
deleteNearZeroCoefficientOfVariation(X, LIMIT = 0.1)
```

Arguments

`X`	Numeric matrix or data.frame. Explanatory variables. Qualitative variables must be transform into binary variables.
`LIMIT`	Numeric. Cutoff for minimum variation. If coefficient is lesser than the limit, the variables are removed because not vary enough (default: 0.1).

Details

The `deleteNearZeroCoefficientOfVariation` function is a pivotal tool in data preprocessing, especially when dealing with high-dimensional datasets. The coefficient of variation (CoV) is a normalized measure of data dispersion, calculated as the ratio of the standard deviation to the mean. In many scientific investigations, variables with a low CoV might be considered as offering limited discriminative information, potentially leading to noise in subsequent statistical analyses. By setting a threshold through the `LIMIT` parameter, this function provides a systematic approach to

identify and exclude variables that do not meet the desired variability criteria. The underlying rationale is that variables with a CoV below the set threshold might not contribute significantly to the variability of the dataset and could be redundant or even detrimental for certain analyses. The function returns a modified dataset, a list of deleted variables, and the computed coefficients of variation for each variable. This comprehensive output ensures that researchers are well-informed about the preprocessing steps and can make subsequent analytical decisions with confidence.

Value

Return a list of two objects: `X`: The new data.frame X filtered. `variablesDeleted`: The variables that have been removed by the filter. `coeff_variation`: The coefficient variables per each variable tested.

Author(s)

Pedro Salguero Garcia. Maintainer: pedsalga@upv.edu.es

Examples

```
data("X_proteomic")
X <- X_proteomic
filter <- deleteNearZeroCoefficientOfVariation(X, LIMIT = 0.1)
```

`deleteNearZeroCoefficientOfVariation.mb`
deleteNearZeroCoefficientOfVariation.mb

Description

Filters out variables from a dataset that exhibit a coefficient of variation below a specified threshold, ensuring the retention of variables with meaningful variability.

Usage

```
deleteNearZeroCoefficientOfVariation.mb(X, LIMIT = 0.1)
```

Arguments

`X`	List of numeric matrices or data.frames. Explanatory variables. Qualitative variables must be transform into binary variables.
`LIMIT`	Numeric. Cutoff for minimum variation. If coefficient is lesser than the limit, the variables are removed because not vary enough (default: 0.1).

Details

The `deleteNearZeroCoefficientOfVariation` function is a pivotal tool in data preprocessing, especially when dealing with high-dimensional datasets. The coefficient of variation (CoV) is a normalized measure of data dispersion, calculated as the ratio of the standard deviation to the mean. In many scientific investigations, variables with a low CoV might be considered as offering limited discriminative information, potentially leading to noise in subsequent statistical analyses. By setting a threshold through the `LIMIT` parameter, this function provides a systematic approach to identify and exclude variables that do not meet the desired variability criteria. The underlying rationale is that variables with a CoV below the set threshold might not contribute significantly to the variability of the dataset and could be redundant or even detrimental for certain analyses. The function returns a modified dataset, a list of deleted variables, and the computed coefficients of variation for each variable. This comprehensive output ensures that researchers are well-informed about the preprocessing steps and can make subsequent analytical decisions with confidence.

Value

A list of three objects. `X`: A list with as many blocks as X input, but with the variables filtered. `variablesDeleted`: A list with as many blocks as X input, with the name of the variables that have been removed. `coeff_variation`: A list with as many blocks as X input, with the coefficient of variation per variable.

Author(s)

Pedro Salguero Garcia. Maintainer: pedsalga@upv.edu.es

Examples

```
data("X_multiomic")
X <- X_multiomic
filter <- deleteNearZeroCoefficientOfVariation.mb(X, LIMIT = 0.1)
```

`deleteZeroOrNearZeroVariance`

deleteZeroOrNearZeroVariance

Description

Provides a robust mechanism to filter out variables from a dataset that exhibit zero or near-zero variance, thereby enhancing the quality and interpretability of subsequent statistical analyses.

Usage

```
deleteZeroOrNearZeroVariance(
  X,
  remove_near_zero_variance = FALSE,
  remove_zero_variance = TRUE,
```

```
  toKeep.zv = NULL,
  freqCut = 95/5
)
```

Arguments

X	Numeric matrix or data.frame. Explanatory variables. Qualitative variables must be transform into binary variables.
remove_near_zero_variance	Logical. If remove_near_zero_variance = TRUE, near zero variance variables will be removed (default: TRUE).
remove_zero_variance	Logical. If remove_zero_variance = TRUE, zero variance variables will be removed (default: TRUE).
toKeep.zv	Character vector. Name of variables in X to not be deleted by (near) zero variance filtering (default: NULL).
freqCut	Numeric. Cutoff for the ratio of the most common value to the second most common value (default: 95/5).

Details

The deleteZeroOrNearZeroVariance function is an indispensable tool in the preprocessing phase of statistical modeling. In many datasets, especially high-dimensional ones, certain variables might exhibit zero or near-zero variance. Such variables can be problematic as they offer limited information variance and can potentially distort the results of statistical models, leading to issues like overfitting. By leveraging the caret::nearZeroVar() function, this tool offers a rigorous method to identify and exclude these variables. Users are afforded flexibility in their choices, with options to remove only zero variance variables, near-zero variance variables, or both. The function also provides the capability to set a frequency cutoff, freqCut, which determines the threshold for near-zero variance based on the ratio of the most frequent value to the second most frequent value. For scenarios where certain variables are deemed essential and should not be removed regardless of their variance, the toKeep.zv parameter allows users to specify a list of such variables.

Value

Return a list of two objects: X: The new data.frame X filtered. variablesDeleted: The variables that have been removed by the filter.

Author(s)

Pedro Salguero Garcia. Maintainer: pedsalga@upv.edu.es

Examples

```
data("X_proteomic")
X <- X_proteomic
filter <- deleteZeroOrNearZeroVariance(X, remove_near_zero_variance = TRUE)
```

deleteZeroOrNearZeroVariance.mb
deleteZeroOrNearZeroVariance.mb

Description

Provides a robust mechanism to filter out variables from a dataset that exhibit zero or near-zero variance, thereby enhancing the quality and interpretability of subsequent statistical analyses.

Usage

```
deleteZeroOrNearZeroVariance.mb(
  X,
  remove_near_zero_variance = FALSE,
  remove_zero_variance = TRUE,
  toKeep.zv = NULL,
  freqCut = 95/5
)
```

Arguments

X	List of numeric matrices or data.frame. Explanatory variables. Qualitative variables must be transform into binary variables.
remove_near_zero_variance	Logical. If remove_near_zero_variance = TRUE, near zero variance variables will be removed (default: TRUE).
remove_zero_variance	Logical. If remove_zero_variance = TRUE, zero variance variables will be removed (default: TRUE).
toKeep.zv	Character vector. Name of variables in X to not be deleted by (near) zero variance filtering (default: NULL).
freqCut	Numeric. Cutoff for the ratio of the most common value to the second most common value (default: 95/5).

Details

The `deleteZeroOrNearZeroVariance` function is an indispensable tool in the preprocessing phase of statistical modeling. In many datasets, especially high-dimensional ones, certain variables might exhibit zero or near-zero variance. Such variables can be problematic as they offer limited information variance and can potentially distort the results of statistical models, leading to issues like overfitting. By leveraging the `caret::nearZeroVar()` function, this tool offers a rigorous method to identify and exclude these variables. Users are afforded flexibility in their choices, with options to remove only zero variance variables, near-zero variance variables, or both. The function also provides the capability to set a frequency cutoff, `freqCut`, which determines the threshold for near-zero variance based on the ratio of the most frequent value to the second most frequent value. For scenarios where certain variables are deemed essential and should not be removed regardless of their variance, the `toKeep.zv` parameter allows users to specify a list of such variables.

Value

A list of two objects. `X`: A list with as many blocks as X input, but with the variables filtered. `variablesDeleted`: A list with as many blocks as X input, with the name of the variables that have been removed.

Author(s)

Pedro Salguero Garcia. Maintainer: pedsalga@upv.edu.es

Examples

```
data("X_multiomic")
X <- X_multiomic
filter <- deleteZeroOrNearZeroVariance.mb(X, remove_near_zero_variance = TRUE)
```

---

eval_Coxmos_models *eval_Coxmos_models*

Description

The `eval_Coxmos_models` function facilitates the comprehensive evaluation of multiple Coxmos models in a concurrent manner. It is designed to provide a detailed assessment of the models' performance by calculating the Area Under the Curve (AUC) for each model at specified time points. The results generated by this function are primed for visualization using the `plot_evaluation()` function.

Usage

```
eval_Coxmos_models(
  lst_models,
  X_test,
  Y_test,
  pred.method = "cenROC",
  pred.attr = "mean",
  times = NULL,
  PARALLEL = FALSE,
  max_time_points = 15,
  verbose = FALSE,
  progress_bar = TRUE
)
```

Arguments

`lst_models`	List of Coxmos models. Each object of the list must be named.
`X_test`	Numeric matrix or data.frame. Explanatory variables for test data (raw format). Qualitative variables must be transform into binary variables.

Y_test	Numeric matrix or data.frame. Response variables for test data. Object must have two columns named as "time" and "event". For event column, accepted values are: 0/1 or FALSE/TRUE for censored and event observations.
pred.method	Character. AUC evaluation algorithm method for evaluate the model performance. Must be one of the following: "risksetROC", "survivalROC", "cenROC", "nsROC", "smoothROCtime_C", "smoothROCtime_I" (default: "cenROC").
pred.attr	Character. Way to evaluate the metric selected. Must be one of the following: "mean" or "median" (default: "mean").
times	Numeric vector. Time points where the AUC will be evaluated. If NULL, a maximum of 'max_time_points' points will be selected equally distributed (default: NULL).
PARALLEL	Logical. Run the cross validation with multicore option. As many cores as your total cores - 1 will be used. It could lead to higher RAM consumption (default: FALSE).
max_time_points	Numeric. Maximum number of time points to use for evaluating the model (default: 15).
verbose	Logical. If verbose = TRUE, extra messages could be displayed (default: FALSE).
progress_bar	Logical. If progress_bar = TRUE, progress bar is shown (default = TRUE).

Details

The function begins by validating the names of the models provided in the `lst_models` list and ensures that there are at least two events present in the dataset. It then checks for the availability of the specified evaluation method and ensures that the test times are consistent with the training times of the models.

The core of the function revolves around the evaluation of each model. Depending on the user's preference, the evaluations can be executed in parallel, which can significantly expedite the process, especially when dealing with a large number of models. The function employs various evaluation methods, as specified by the `pred.method` parameter, to compute the AUC values. These methods include but are not limited to "risksetROC", "survivalROC", and "cenROC".

Post-evaluation, the function collates the results, including training times, AIC values, c-index, Brier scores, and AUC values for each time point. The results are then transformed into a structured data frame, making it conducive for further analysis and visualization. It's worth noting that potential issues in AUC computation, often arising from sparse samples, are flagged to the user for further inspection.

Value

A list of four objects. `df`: A data.frame which the global predictions for all models. This data.frame is used to plot the information by the function `plot_evaluation()`. `lst_AUC`: A list of models where the user can check the linear predictors computed, the global AUC, the AUC per time point and the predicted time points selected. `lst_BRIER`: A list of models where the user can check the predicted time points selected, the Brier Score per time point and the Integrative Brier score (computed by `survcomp::sbrier.score2proba`). `time`: Time used for evaluation process.

Author(s)

Pedro Salguero Garcia. Maintainer: pedsalga@upv.edu.es

References

Harrell FE, Califf RM, Pryor DB, Lee KL, Rosati RA (1982). "Evaluating the Yield of Medical Tests." *JAMA*, **247**. doi:10.1001/jama.1982.03320430047030, https://jamanetwork.com/journals/jama. MS S, AC C, J Q, B H (2011). "survcomp: an R/Bioconductor package for performance assessment and comparison of survival models." *Bioinformatics*, **27(22)**, 3206-3208. Heagerty PJ, Lumley T, Pepe MS (2000). "Time-Dependent ROC Curves for Censored Survival Data and a Diagnostic Marker." *Biometrics*. Heagerty PJ, Zheng Y (2005). "Survival Model Predictive Accuracy and ROC Curves." *Biometrics*, **61**, 92-105. doi:10.1111/j.0006341x.2005.030814.x. Beyene KM, Ghouch AE (2020). "Smoothed time-dependent receiver operating characteristic curve for right censored survival data." *Statistics in Medicine*, **39**(24), 3373-3396. ISSN 10970258, https://pubmed.ncbi.nlm.nih.gov/32687225/. Pérez-Fernández S, Martínez-Camblor P, Filzmoser P, Corral N (2018). "nsROC: An R package for Non-Standard ROC Curve Analysis." *The R Journal*. doi:10.1007/s00180020009557. Díaz-Coto S, Martínez-Camblor P, Pérez-Fernández S (2020). "smoothROCtime: an R package for time-dependent ROC curve estimation." *Computational Statistics*, **35**(3), 1231-1251. ISSN 16139658, doi:10.1007/s00180020009557.

Examples

```
data("X_proteomic")
data("Y_proteomic")
set.seed(123)
index_train <- caret::createDataPartition(Y_proteomic$event, p = .5, list = FALSE, times = 1)
X_train <- X_proteomic[index_train,1:50]
Y_train <- Y_proteomic[index_train,]

X_test <- X_proteomic[-index_train,1:50]
Y_test <- Y_proteomic[-index_train,]

model_icox <- splsicox(X_train, Y_train, n.comp = 2)
model_drcox <- splsdrcox(X_train, Y_train, n.comp = 2)
lst_models <- list("splsicox" = model_icox, "splsdrcox" = model_drcox)
eval_Coxmos_models(lst_models, X_test, Y_test, pred.method = "cenROC")
```

`eval_Coxmos_model_per_variable`

eval_Coxmos_model_per_variable

Description

The `eval_Coxmos_model_per_variable` function offers a granular evaluation of a specific Coxmos model, focusing on the influence of individual variables or components on the model's predictive performance. It computes the Area Under the Curve (AUC) for each variable at designated time points, providing insights into the relative importance of each variable in the model's predictions. For a visual representation of the results, it is advisable to utilize the `plot_evaluation()` function post-evaluation.

Usage

```
eval_Coxmos_model_per_variable(
  model,
  X_test,
  Y_test,
  pred.method = "cenROC",
  pred.attr = "mean",
  times = NULL,
  max_time_points = 15,
  PARALLEL = FALSE,
  verbose = FALSE
)
```

Arguments

model	Coxmos model.
X_test	Numeric matrix or data.frame. Explanatory variables for test data (raw format). Qualitative variables must be transform into binary variables.
Y_test	Numeric matrix or data.frame. Response variables for test data. Object must have two columns named as "time" and "event". For event column, accepted values are: 0/1 or FALSE/TRUE for censored and event observations.
pred.method	Character. AUC evaluation algorithm method for evaluate the model performance. Must be one of the following: "risksetROC", "survivalROC", "cenROC", "nsROC", "smoothROCtime_C", "smoothROCtime_I" (default: "cenROC").
pred.attr	Character. Way to evaluate the metric selected. Must be one of the following: "mean" or "median" (default: "mean").
times	Numeric vector. Time points where the AUC will be evaluated. If NULL, a maximum of 'max_time_points' points will be selected equally distributed (default: NULL).
max_time_points	Numeric. Maximum number of time points to use for evaluating the model (default: 15).
PARALLEL	Logical. Run the cross validation with multicore option. As many cores as your total cores - 1 will be used. It could lead to higher RAM consumption (default: FALSE).
verbose	Logical. If verbose = TRUE, extra messages could be displayed (default: FALSE).

Details

Upon invocation, the function initiates by verifying the consistency between test times and the training times of the provided model. Subsequently, linear predictors for each variable are derived using the `predict.Coxmos` function. These linear predictors serve as the foundation for the AUC computation, which is executed for each variable across the specified time points.

The function employs various evaluation methods, as determined by the `pred.method` parameter, to calculate the AUC values. These methods encompass options such as "risksetROC", "survivalROC", and "cenROC", among others. The results are systematically organized into a structured data

frame, segregating AUC values for each variable at different time points. This structured output not only facilitates easy interpretation but also sets the stage for subsequent visualization or further analysis.

It's noteworthy that the function is equipped to handle parallel processing, contingent on the user's preference, which can expedite the evaluation process, especially when dealing with extensive datasets or multiple time points.

Value

A list of two objects: `df`: A data.frame which the predictions for the specific model split into the full model (LP) and each component individually. This data.frame is used to plot the information by the function `plot_evaluation()`. `lst_AUC`: A list of the full model prediction and its components where the user can check the linear predictors used, the global AUC, the AUC per time point and the predicted time points selected.

Author(s)

Pedro Salguero Garcia. Maintainer: pedsalga@upv.edu.es

Examples

```
data("X_proteomic")
data("Y_proteomic")
set.seed(123)
index_train <- caret::createDataPartition(Y_proteomic$event, p = .5, list = FALSE, times = 1)
X_train <- X_proteomic[index_train,1:50]
Y_train <- Y_proteomic[index_train,]

X_test <- X_proteomic[-index_train,1:50]
Y_test <- Y_proteomic[-index_train,]

model_icox <- splsicox(X_train, Y_train, n.comp = 2)
eval_Coxmos_model_per_variable(model_icox, X_test, Y_test, pred.method = "cenROC")
```

---

factorToBinary *factorToBinary*

Description

Transforms factor variables within a matrix or data frame into binary dummy variables, facilitating numerical representation for subsequent statistical analyses. The function provides an option to generate either k or k-1 dummy variables for each factor, contingent on its levels.

Usage

```
factorToBinary(X, all = TRUE, sep = "_")
```

Arguments

X	Numeric matrix or data.frame. Only qualitative variables (factor class) will be transformed into binary variables.
all	Logical. If all = TRUE, as many variables as levels will be returned in the new matrix. Otherwise, k-1 variables will be used where the first level will be use as "default" state (default: TRUE).
sep	Character. Character symbol to generate new colnames. Ex. If variable name is "sex" and sep = "_". Dummy variables will be "sex_male" and "sex_female".

Details

The `factorToBinary` function addresses a recurrent challenge in data preprocessing: the conversion of factor variables into a numerical format suitable for a plethora of statistical and machine learning algorithms. Factors, inherently categorical in nature, often necessitate transformation into a binary format, commonly referred to as dummy or one-hot encoding. This function adeptly performs this transformation, iterating over each column of the provided matrix or data frame. When encountering factor variables, it employs the `model.matrix` function to generate the requisite dummy variables. The user's discretion is paramount in determining the number of dummy variables: either k, equivalent to the number of levels for the factor, or k-1, where the omitted level serves as a reference or "default" state. This choice is particularly salient in regression contexts to circumvent multicollinearity issues. The naming convention for the resultant dummy variables amalgamates the original factor's name with its respective level, separated by a user-defined character, ensuring clarity and interpretability. Non-factor variables remain unaltered, preserving the integrity of the original data structure.

Value

A matrix or data.frame with k-1 or k dummy variables for categorical/factor data.

Author(s)

Pedro Salguero Garcia. Maintainer: pedsalga@upv.edu.es

Examples

```
data("X_proteomic")
X <- X_proteomic
X.dummy <- factorToBinary(X, all = FALSE, sep = "_")
X.pls <- factorToBinary(X, all = TRUE, sep = "_")
```

---

getAutoKM *getAutoKM*

Description

Generates a Kaplan-Meier plot for the specified Coxmos model. The plot can be constructed based on the model's Linear Predictor value, the PLS-COX component, or the original variable level.

Usage

```
getAutoKM(
  type = "LP",
  model,
  comp = 1:2,
  top = 10,
  ori_data = TRUE,
  BREAKTIME = NULL,
  n.breaks = 20,
  only_sig = FALSE,
  alpha = 0.05,
  title = NULL,
  verbose = FALSE
)
```

Arguments

type	Character. Kaplan Meier for complete model linear predictor ("LP"), for PLS components ("COMP") or for original variables ("VAR") (default: LP).
model	Coxmos model.
comp	Numeric vector. Vector of length two. Select which components to plot (default: c(1,2)).
top	Numeric. Show "top" first variables. If top = NULL, all variables are shown (default: 10).
ori_data	Logical. Compute the Kaplan-Meier plot with the raw-data or the normalize-data to compute the best cut-point for splitting the data into two groups. Only used when type = "VAR" (default: TRUE).
BREAKTIME	Numeric. Size of time to split the data into "total_time / BREAKTIME + 1" points. If BREAKTIME = NULL, "n.breaks" is used (default: NULL).
n.breaks	Numeric. If BREAKTIME is NULL, "n.breaks" is the number of time-break points to compute (default: 20).
only_sig	Logical. If "only_sig" = TRUE, then only significant log-rank test variables are returned (default: FALSE).
alpha	Numeric. Numerical values are regarded as significant if they fall below the threshold (default: 0.05).
title	Character. Kaplan-Meier plot title (default: NULL).
verbose	Logical. If verbose = TRUE, extra messages could be displayed (default: FALSE).

Details

The `getAutoKM` function offers a flexible approach to visualize survival analysis results using the Kaplan-Meier method. Depending on the `type` parameter, the function can generate plots based on different aspects of the Coxmos model:

- "LP": Uses the Linear Predictor value of the model.

- "COMP": Utilizes the PLS-COX component.
- "VAR": Operates at the original variable level.

The function provides options to customize the number of components (`comp`), the number of top variables (`top`), and whether to use raw or normalized data (`ori_data`). Additionally, users can specify the time intervals (`BREAKTIME` and `n.breaks`) for the Kaplan-Meier plot. If significance testing is desired, the function can filter out non-significant variables based on the log-rank test (`only_sig` and `alpha` parameters).

It's essential to ensure that the provided `model` is of the correct class (`Coxmos`). The function will return an error message if an incompatible model is supplied.

Value

A list of two elements per each model in the list: `info_logrank_num`: A list of two data.frames with the numerical variables categorize as qualitative and the cutpoint to divide the data into two groups. `LST_PLOTS`: A list with the Kaplan-Meier Plots.

Author(s)

Pedro Salguero Garcia. Maintainer: pedsalga@upv.edu.es

References

Kaplan EL, Kaplan EL, Meier P (1958). "Nonparametric Estimation from Incomplete Observations." *Journal of the American Statistical Association*. doi:10.1007/9781461243809_25, https://link.springer.com/chapter/10.1007/978-1-4612-4380-9_25.

Examples

```
data("X_proteomic")
data("Y_proteomic")
set.seed(123)
index_train <- caret::createDataPartition(Y_proteomic$event, p = .5, list = FALSE, times = 1)
X_train <- X_proteomic[index_train,1:50]
Y_train <- Y_proteomic[index_train,]
X_test <- X_proteomic[-index_train,1:50]
Y_test <- Y_proteomic[-index_train,]
splsicox.model <- splsicox(X_train, Y_train, n.comp = 2, penalty = 0.5, x.center = TRUE,
x.scale = TRUE)
getAutoKM(type = "LP", model = splsicox.model)
```

`getAutoKM.list` *getAutoKM.list*

Description

Run the function "getAutoKM" for a list of models. More information in "?getAutoKM".

Usage

```
getAutoKM.list(
  type = "LP",
  lst_models,
  comp = 1:2,
  top = NULL,
  ori_data = TRUE,
  BREAKTIME = NULL,
  n.breaks = 20,
  only_sig = FALSE,
  alpha = 0.05,
  title = NULL,
  verbose = FALSE
)
```

Arguments

type	Character. Kaplan Meier for complete model linear predictor ("LP"), for PLS components ("COMP") or for original variables ("VAR") (default: LP).
lst_models	List of Coxmos models.
comp	Numeric vector. Vector of length two. Select which components to plot (default: c(1,2)).
top	Numeric. Show "top" first variables. If top = NULL, all variables are shown (default: 10).
ori_data	Logical. Compute the Kaplan-Meier plot with the raw-data or the normalize-data to compute the best cut-point for splitting the data into two groups. Only used when type = "VAR" (default: TRUE).
BREAKTIME	Numeric. Size of time to split the data into "total_time / BREAKTIME + 1" points. If BREAKTIME = NULL, "n.breaks" is used (default: NULL).
n.breaks	Numeric. If BREAKTIME is NULL, "n.breaks" is the number of time-break points to compute (default: 20).
only_sig	Logical. If "only_sig" = TRUE, then only significant log-rank test variables are returned (default: FALSE).
alpha	Numeric. Numerical values are regarded as significant if they fall below the threshold (default: 0.05).
title	Character. Kaplan-Meier plot title (default: NULL).
verbose	Logical. If verbose = TRUE, extra messages could be displayed (default: FALSE).

Value

A list of two elements per each model in the list: `info_logrank_num`: A list of two data.frames with the numerical variables categorize as qualitative and the cutpoint to divide the data into two groups. `LST_PLOTS`: A list with the Kaplan-Meier Plots.

Author(s)

Pedro Salguero Garcia. Maintainer: pedsalga@upv.edu.es

References

Kaplan EL, Kaplan EL, Meier P (1958). "Nonparametric Estimation from Incomplete Observations." *Journal of the American Statistical Association*. doi:10.1007/9781461243809_25, https://link.springer.com/chapter/10.1007/978-1-4612-4380-9_25.

Examples

```
data("X_proteomic")
data("Y_proteomic")
set.seed(123)
index_train <- caret::createDataPartition(Y_proteomic$event, p = .5, list = FALSE, times = 1)
X_train <- X_proteomic[index_train,1:20]
Y_train <- Y_proteomic[index_train,]
X_test <- X_proteomic[-index_train,1:20]
Y_test <- Y_proteomic[-index_train,]
splsicox.model <- splsicox(X_train, Y_train, n.comp = 1, penalty = 0.5, x.center = TRUE,
x.scale = TRUE)
splsdrcox.model <- splsdrcox(X_train, Y_train, n.comp = 1, penalty = 0.5, x.center = TRUE,
x.scale = TRUE)
lst_models = list("sPLSICOX" = splsicox.model, "sPLSDRCOX" = splsdrcox.model)
getAutoKM.list(type = "LP", lst_models)
```

---

getCutoffAutoKM *getCutoffAutoKM*

Description

Gets the cutoff value from the results of getAutoKM() functions.

Usage

```
getCutoffAutoKM(result)
```

Arguments

result	List. Result of getAutoKM() function.

Value

A named numeric vector where each element represents the cutoff value.

Author(s)

Pedro Salguero Garcia. Maintainer: pedsalga@upv.edu.es

References

Kaplan EL, Kaplan EL, Meier P (1958). "Nonparametric Estimation from Incomplete Observations." *Journal of the American Statistical Association*. doi:10.1007/9781461243809_25, https://link.springer.com/chapter/10.1007/978-1-4612-4380-9_25.

Examples

```
data("X_proteomic")
data("Y_proteomic")
set.seed(123)
index_train <- caret::createDataPartition(Y_proteomic$event, p = .5, list = FALSE, times = 1)
X_train <- X_proteomic[index_train,1:50]
Y_train <- Y_proteomic[index_train,]
X_test <- X_proteomic[-index_train,1:50]
Y_test <- Y_proteomic[-index_train,]
splsicox.model <- splsicox(X_train, Y_train, n.comp = 2, penalty = 0.5, x.center = TRUE,
x.scale = TRUE)
KMresult = getAutoKM(type = "LP", model = splsicox.model)
getCutoffAutoKM(result = KMresult)
```

---

getCutoffAutoKM.list *getCutoffAutoKM.list*

Description

Run the function "getCutoffAutoKM" for a list of models. More information in "?getCutoffAutoKM".

Usage

```
getCutoffAutoKM.list(lst_results)
```

Arguments

`lst_results`	List of lists. Result of getAutoKM.list() function.

Value

A list where each element corresponds to the result of the `getCutoffAutoKM` function applied to each model in the input list. The structure and content of each element will be consistent with the output of the `getCutoffAutoKM` function.

Author(s)

Pedro Salguero Garcia. Maintainer: pedsalga@upv.edu.es

References

Kaplan EL, Kaplan EL, Meier P (1958). "Nonparametric Estimation from Incomplete Observations." *Journal of the American Statistical Association*. doi:10.1007/9781461243809_25, https://link.springer.com/chapter/10.1007/978-1-4612-4380-9_25.

Examples

```
data("X_proteomic")
data("Y_proteomic")
set.seed(123)
index_train <- caret::createDataPartition(Y_proteomic$event, p = .5, list = FALSE, times = 1)
X_train <- X_proteomic[index_train,1:50]
Y_train <- Y_proteomic[index_train,]
X_test <- X_proteomic[-index_train,1:50]
Y_test <- Y_proteomic[-index_train,]
splsicox.model <- splsicox(X_train, Y_train, n.comp = 2, penalty = 0.5, x.center = TRUE,
x.scale = TRUE)
splsdrcox.model <- splsdrcox(X_train, Y_train, n.comp = 2, penalty = 0.5, x.center = TRUE,
x.scale = TRUE)
lst_models = list("sPLSICOX" = splsicox.model, "sPLSDRCOX" = splsdrcox.model)
lst_results = getAutoKM.list(type = "LP", lst_models)
getCutoffAutoKM.list(lst_results)
```

---

getEPV *getEPV*

Description

Provides a quantitative assessment of the dataset by computing the Events per Variable (EPV) metric, which gauges the proportionality between observed events and the number of explanatory variables.

Usage

```
getEPV(X, Y)
```

Arguments

X	Numeric matrix or data.frame. Explanatory variables. Qualitative variables must be transform into binary variables.
Y	Numeric matrix or data.frame. Response variables. Object must have two columns named as "time" and "event". For event column, accepted values are: 0/1 or FALSE/TRUE for censored and event observations.

Details

In the realm of survival analysis, the balance between observed events and explanatory variables is paramount. The `getEPV` function serves as a tool for researchers to ascertain this balance, which can be pivotal in determining the robustness and interpretability of subsequent statistical models. By evaluating the ratio of events in the `Y` matrix to the variables in the `X` matrix, the function yields the EPV metric. It is of utmost importance that the `Y` matrix encompasses two distinct columns, namely "time" and "event". The latter, "event", should strictly encapsulate binary values, delineating censored (either 0 or FALSE) and event (either 1 or TRUE) observations. To ensure the integrity of the data and the precision of the computation, the function is equipped with an error mechanism that activates if the "event" column remains undetected.

Value

Return the EPV value for a specific X (explanatory variables) and Y (time and censored variables) data.

Author(s)

Pedro Salguero Garcia. Maintainer: pedsalga@upv.edu.es

Examples

```
data("X_proteomic")
data("Y_proteomic")
X <- X_proteomic
Y <- Y_proteomic
getEPV(X,Y)
```

`getEPV.mb` *getEPV.mb*

Description

Provides a quantitative assessment of the dataset by computing the Events per Variable (EPV) metric for multi-block data, which gauges the proportionality between observed events and the number of explanatory variables.

Usage

```
getEPV.mb(X, Y)
```

Arguments

`X`	Numeric matrix or data.frame. Explanatory variables. Qualitative variables must be transform into binary variables.
`Y`	Numeric matrix or data.frame. Response variables. Object must have two columns named as "time" and "event". For event column, accepted values are: 0/1 or FALSE/TRUE for censored and event observations.

Details

In the realm of survival analysis, the balance between observed events and explanatory variables is paramount. The `getEPV` function serves as a tool for researchers to ascertain this balance, which can be pivotal in determining the robustness and interpretability of subsequent statistical models. By evaluating the ratio of events in the `Y` matrix to the variables in the `X` matrix, the function yields the EPV metric. It is of utmost importance that the `Y` matrix encompasses two distinct columns, namely "time" and "event". The latter, "event", should strictly encapsulate binary values, delineating censored (either 0 or FALSE) and event (either 1 or TRUE) observations. To ensure the integrity of the data and the precision of the computation, the function is equipped with an error mechanism that activates if the "event" column remains undetected.

Value

Return the EPV value for a specific X (explanatory variables) and Y (time and censored variables) data.

Author(s)

Pedro Salguero Garcia. Maintainer: pedsalga@upv.edu.es

Examples

```
data("X_multiomic")
data("Y_multiomic")
X <- X_multiomic
Y <- Y_multiomic
getEPV.mb(X,Y)
```

`getTestKM` *getTestKM*

Description

This function computes and visualizes the Kaplan-Meier survival curve for a given test dataset, utilizing the cutoff derived from the original model. The function offers flexibility in terms of the type of Kaplan-Meier estimation, whether it's based on the linear predictor, PLS components, or original variables.

Usage

```
getTestKM(
  model,
  X_test,
  Y_test,
  cutoff,
  type = "LP",
  ori_data = TRUE,
  BREAKTIME = NULL,
  n.breaks = 20,
  title = NULL
)
```

Arguments

`model`	Coxmos model.
`X_test`	Numeric matrix or data.frame. Explanatory variables for test data (raw format). Qualitative variables must be transform into binary variables.

Y_test	Numeric matrix or data.frame. Response variables for test data. Object must have two columns named as "time" and "event". For event column, accepted values are: 0/1 or FALSE/TRUE for censored and event observations.
cutoff	Numeric. Cutoff value to split the observations into two groups. Recommended to compute optimal cutoff value with getAutoKM() function.
type	Character. Kaplan Meier for complete model linear predictor ("LP"), for PLS components ("COMP") or for original variables ("VAR") (default: LP).
ori_data	Logical. Compute the Kaplan-Meier plot with the raw-data or the normalize-data to compute the best cut-point for splitting the data into two groups. Only used when type = "VAR" (default: TRUE).
BREAKTIME	Numeric. Size of time to split the data into "total_time / BREAKTIME + 1" points. If BREAKTIME = NULL, "n.breaks" is used (default: NULL).
n.breaks	Numeric. If BREAKTIME is NULL, "n.breaks" is the number of time-break points to compute (default: 20).
title	Character. Kaplan-Meier plot title (default: NULL).

Details

The getTestKM function is designed to evaluate the survival probabilities of a test dataset based on a pre-trained Coxmos model. The function ensures that the test times are consistent with the training times. Depending on the specified type, the function can compute the Kaplan-Meier curve using:

- The complete model's linear predictor (LP).
- The PLS components (COMP).
- The original variables (VAR).

For the LP type, the function predicts scores for the X_test and subsequently predicts the linear predictor using these scores. For the COMP type, the function predicts scores for each component in the model and computes the Kaplan-Meier curve for each. For the VAR type, the function computes the Kaplan-Meier curve for each variable in the test dataset.

The function also provides the flexibility to compute the Kaplan-Meier plot using raw data or normalized data, which can be useful for determining the optimal cut-point for data segmentation. The time intervals for the Kaplan-Meier estimation can be defined using either the BREAKTIME or n.breaks parameters.

The resulting Kaplan-Meier plot provides a visual representation of the survival probabilities over time, segmented based on the specified cutoff. This allows for a comprehensive evaluation of the test dataset's survival characteristics in the context of the original model.

Value

Depending on the specified type parameter, the function returns:

- LP: A ggplot object visualizing the Kaplan-Meier survival curve based on the linear predictor, segmented by the specified cutoff.
- COMP: A list of ggplot objects, where each plot represents the Kaplan-Meier survival curve for a specific PLS component in the model, segmented by the respective cutoffs.

- VAR: A list of ggplot objects, where each plot visualizes the Kaplan-Meier survival curve for a specific variable in the test dataset, segmented by the respective cutoffs.

Each plot provides a visual representation of the survival probabilities over time, allowing for a comprehensive evaluation of the test dataset's survival characteristics in the context of the original model.

Author(s)

Pedro Salguero Garcia. Maintainer: pedsalga@upv.edu.es

References

Kaplan EL, Kaplan EL, Meier P (1958). "Nonparametric Estimation from Incomplete Observations." *Journal of the American Statistical Association*. doi:10.1007/9781461243809_25, https://link.springer.com/chapter/10.1007/978-1-4612-4380-9_25.

Examples

```
data("X_proteomic")
data("Y_proteomic")
set.seed(123)
index_train <- caret::createDataPartition(Y_proteomic$event, p = .5, list = FALSE, times = 1)
X_train <- X_proteomic[index_train,1:50]
Y_train <- Y_proteomic[index_train,]
X_test <- X_proteomic[-index_train,1:50]
Y_test <- Y_proteomic[-index_train,]
splsicox.model <- splsicox(X_train, Y_train, n.comp = 2, penalty = 0.5, x.center = TRUE,
x.scale = TRUE)
KMresult = getAutoKM(type = "LP", model = splsicox.model)
cutoff <- getCutoffAutoKM(result = KMresult)
getTestKM(splsicox.model, X_test, Y_test, cutoff)
```

`getTestKM.list` *getTestKM.list*

Description

Run the function "getTestKM" for a list of models. More information in "?getTestKM".

Usage

```
getTestKM.list(
  lst_models,
  X_test,
  Y_test,
  lst_cutoff,
  type = "LP",
  ori_data = TRUE,
```

```
  BREAKTIME = NULL,
  n.breaks = 20,
  title = NULL,
  verbose = FALSE
)
```

Arguments

lst_models	List of Coxmos model
X_test	Numeric matrix or data.frame. Explanatory variables for test data (raw format). Qualitative variables must be transform into binary variables.
Y_test	Numeric matrix or data.frame. Response variables for test data. Object must have two columns named as "time" and "event". For event column, accepted values are: 0/1 or FALSE/TRUE for censored and event observations.
lst_cutoff	Numeric vector. Cutoff vector to split the observations into two groups for each model. Recommended to compute optimal cutoff value with getAutoKM() or getAutoKM.list() functions.
type	Character. Kaplan Meier for complete model linear predictor ("LP"), for PLS components ("COMP") or for original variables ("VAR") (default: LP).
ori_data	Logical. Compute the Kaplan-Meier plot with the raw-data or the normalize-data to compute the best cut-point for splitting the data into two groups. Only used when type = "VAR" (default: TRUE).
BREAKTIME	Numeric. Size of time to split the data into "total_time / BREAKTIME + 1" points. If BREAKTIME = NULL, "n.breaks" is used (default: NULL).
n.breaks	Numeric. If BREAKTIME is NULL, "n.breaks" is the number of time-break points to compute (default: 20).
title	Character. Kaplan-Meier plot title (default: NULL).
verbose	Logical. If verbose = TRUE, extra messages could be displayed (default: FALSE).

Value

A list where each element corresponds to a Kaplan-Meier plot generated for each model in the input list. Each plot visualizes the survival probabilities based on the specified cutoff values for the respective model. The list's names correspond to the names of the models provided in the input list.

Author(s)

Pedro Salguero Garcia. Maintainer: pedsalga@upv.edu.es

References

Kaplan EL, Kaplan EL, Meier P (1958). "Nonparametric Estimation from Incomplete Observations." *Journal of the American Statistical Association*. doi:10.1007/9781461243809_25, https://link.springer.com/chapter/10.1007/978-1-4612-4380-9_25.

Examples

```
data("X_proteomic")
data("Y_proteomic")
X_proteomic <- X_proteomic[1:50,]
Y_proteomic <- Y_proteomic[1:50,]
set.seed(123)
index_train <- caret::createDataPartition(Y_proteomic$event, p = .5, list = FALSE, times = 1)
X_train <- X_proteomic[index_train,1:20]
Y_train <- Y_proteomic[index_train,]
X_test <- X_proteomic[-index_train,1:20]
Y_test <- Y_proteomic[-index_train,]
splsicox.model <- splsicox(X_train, Y_train, n.comp = 1, penalty = 0.5, x.center = TRUE,
x.scale = TRUE)
splsdrcox.model <- splsdrcox(X_train, Y_train, n.comp = 1, penalty = 0.5, x.center = TRUE,
x.scale = TRUE)
lst_models = list("sPLSICOX" = splsicox.model, "sPLSDRCOX" = splsdrcox.model)
lst_results = getAutoKM.list(type = "LP", lst_models)
lst_cutoff <- getCutoffAutoKM.list(lst_results)
getTestKM.list(lst_models, X_test, Y_test, lst_cutoff)
```

`loadingplot.Coxmos` *loadingplot.Coxmos*

Description

The `loadingplot.Coxmos` function visualizes the loading values of a given Coxmos model. The function produces a series of bar plots for each component's loading values, offering a comprehensive view of the model's variable contributions. The plots can be customized to exclude zero loadings, display only the top variables, and automatically adjust the color scale limits.

Usage

```
loadingplot.Coxmos(model, zero.rm = TRUE, top = NULL, auto.limits = TRUE)
```

Arguments

`model`	Coxmos model.
`zero.rm`	Logical. Remove variables equal to 0 (default: TRUE).
`top`	Numeric. Show "top" first variables. If top = NULL, all variables are shown (default: NULL).
`auto.limits`	Logical. If "auto.limits" = TRUE, limits are detected automatically (default: TRUE).

Details

The primary objective of the `loadingplot.Coxmos` function is to facilitate the interpretation of Coxmos models by visualizing the loading values of each component. The function first verifies the class of the provided model to ensure it is a valid Coxmos model.

The loading values are extracted from the model and processed based on the user's specifications. If the `zero.rm` parameter is set to TRUE, variables with zero loadings are excluded from the visualization. Additionally, if the `top` parameter is specified, only the top variables, ranked by their absolute loading values, are displayed.

The function employs the 'ggplot2' framework for visualization. The color scale of the plots can be automatically adjusted based on the maximum absolute loading value when `auto.limits` is set to TRUE. If the `RColorConesa` package is available, it utilizes its color palettes for enhanced visualization; otherwise, default colors are applied.

Value

A list of `ggplot2` objects, each representing the loading values for a component of the Coxmos model.

Examples

```
data("X_proteomic")
data("Y_proteomic")
X <- X_proteomic[,1:50]
Y <- Y_proteomic
splsicox.model <- splsicox(X, Y, n.comp = 2, penalty = 0.5, x.center = TRUE, x.scale = TRUE)
loadingplot.Coxmos(model = splsicox.model)
```

`loadingplot.fromVector.Coxmos`

loadingplot.fromVector.Coxmos

Description

loadingplot.fromVector.Coxmos

Usage

```
loadingplot.fromVector.Coxmos(
  model,
  vector,
  zero.rm = FALSE,
  top = NULL,
  auto.limits = TRUE
)
```

Arguments

model	Coxmos model.
vector	Vector of loading
zero.rm	Logical. Remove variables equal to 0 (default: FALSE).
top	Numeric. Show "top" first variables. If top = NULL, all variables are shown (default: NULL).
auto.limits	Logical. If "auto.limits" = TRUE, limits are detected automatically (default: TRUE).

---

mb.splsdacox *MB.sPLS-DACOX*

Description

The MB.sPLS-DACOX function conducts a multi-block sparse partial least squares discriminant analysis Cox (MB.sPLS-DACOX) using a dynamic variable selection approach. This analysis is particularly suited for high-dimensional datasets where the goal is to identify the relationship between explanatory variables and survival outcomes. The function outputs a model of class "Coxmos" with an attribute labeled "MB.sPLS-DACOX".

Usage

```
mb.splsdacox(
  X,
  Y,
  n.comp = 4,
  vector = NULL,
  MIN_NVAR = 10,
  MAX_NVAR = 10000,
  n.cut_points = 5,
  EVAL_METHOD = "AUC",
  x.center = TRUE,
  x.scale = FALSE,
  remove_near_zero_variance = TRUE,
  remove_zero_variance = TRUE,
  toKeep.zv = NULL,
  remove_non_significant = TRUE,
  alpha = 0.05,
  MIN_AUC_INCREASE = 0.01,
  pred.method = "cenROC",
  max.iter = 200,
  times = NULL,
  max_time_points = 15,
  MIN_EPV = 5,
  returnData = TRUE,
  verbose = FALSE
)
```

Arguments

X	Numeric matrix or data.frame. Explanatory variables. Qualitative variables must be transform into binary variables.
Y	Numeric matrix or data.frame. Response variables. Object must have two columns named as "time" and "event". For event column, accepted values are: 0/1 or FALSE/TRUE for censored and event observations.
n.comp	Numeric. Number of latent components to compute for the (s)PLS model (default: 10).
vector	Numeric vector or list. Used for computing best number of variables. As many values as components have to be provided. If vector = NULL, an automatic detection is perform (default: NULL). If vector is a list, must be named as the names of X param followed by the number of variables to select.
MIN_NVAR	Numeric. Minimum range size for computing cut points to select the best number of variables to use (default: 10).
MAX_NVAR	Numeric. Maximum range size for computing cut points to select the best number of variables to use (default: 1000).
n.cut_points	Numeric. Number of cut points for searching the optimal number of variables. If only two cut points are selected, minimum and maximum size are used. For MB approaches as many as n.cut_points^n.blocks models will be computed as minimum (default: 5).
EVAL_METHOD	Character. If EVAL_METHOD = "AUC", AUC metric will be use to compute the best number of variables. In other case, c-index metric will be used (default: "AUC").
x.center	Logical. If x.center = TRUE, X matrix is centered to zero means (default: TRUE).
x.scale	Logical. If x.scale = TRUE, X matrix is scaled to unit variances (default: FALSE).
remove_near_zero_variance	Logical. If remove_near_zero_variance = TRUE, near zero variance variables will be removed (default: TRUE).
remove_zero_variance	Logical. If remove_zero_variance = TRUE, zero variance variables will be removed (default: TRUE).
toKeep.zv	Character vector. Name of variables in X to not be deleted by (near) zero variance filtering (default: NULL).
remove_non_significant	Logical. If remove_non_significant = TRUE, non-significant variables/components in final cox model will be removed until all variables are significant by forward selection (default: FALSE).
alpha	Numeric. Numerical values are regarded as significant if they fall below the threshold (default: 0.05).
MIN_AUC_INCREASE	Numeric. Minimum improvement between different cross validation models to continue evaluating higher values in the multiple tested parameters. If it is not reached for next 'MIN_COMP_TO_CHECK' models and the minimum 'MIN_AUC' value is reached, the evaluation stops (default: 0.01).

pred.method	Character. AUC evaluation algorithm method for evaluate the model performance. Must be one of the following: "risksetROC", "survivalROC", "cenROC", "nsROC", "smoothROCtime_C", "smoothROCtime_I" (default: "cenROC").
max.iter	Numeric. Maximum number of iterations for PLS convergence (default: 200).
times	Numeric vector. Time points where the AUC will be evaluated. If NULL, a maximum of 'max_time_points' points will be selected equally distributed (default: NULL).
max_time_points	Numeric. Maximum number of time points to use for evaluating the model (default: 15).
MIN_EPV	Numeric. Minimum number of Events Per Variable (EPV) you want reach for the final cox model. Used to restrict the number of variables/components can be computed in final cox models. If the minimum is not meet, the model cannot be computed (default: 5).
returnData	Logical. Return original and normalized X and Y matrices (default: TRUE).
verbose	Logical. If verbose = TRUE, extra messages could be displayed (default: FALSE).

Details

The MB.sPLS-DACOX methodology is designed to handle multi-block datasets, where each block represents a set of related variables. By employing a sparse partial least squares approach, the function efficiently selects relevant variables from each block, ensuring that the final model is both interpretable and predictive. The Cox proportional hazards model is then applied to the selected variables to assess their association with survival outcomes.

The function offers flexibility in terms of parameter tuning. For instance, users can specify the number of latent components to compute, the range of variables to consider for optimal selection, and the evaluation metric (either AUC or c-index). Additionally, data preprocessing options are available, such as centering and scaling of the explanatory variables, and removal of variables with near-zero or zero variance.

Value

Instance of class "Coxmos" and model "MB.sPLS-DACOX". The class contains the following elements: X: List of normalized X data information.

- (data): normalized X matrix
- (weightings): PLS weights
- (weightings_norm): PLS normalize weights
- (W.star): PLS W* vector
- (scores): PLS scores/variates
- (E): error matrices
- (x.mean): mean values for X matrix
- (x.sd): standard deviation for X matrix

Y: List of normalized Y data information.

- (deviance_residuals): deviance residual vector used as Y matrix in the sPLS.
- (dr.mean): mean values for deviance residuals Y matrix
- (dr.sd): standard deviation for deviance residuals Y matrix'
- (data): normalized X matrix
- (y.mean): mean values for Y matrix
- (y.sd): standard deviation for Y matrix'

survival_model: List of survival model information.

- fit: coxph object.
- AIC: AIC of cox model.
- BIC: BIC of cox model.
- lp: linear predictors for train data.
- coef: Coefficients for cox model.
- YChapeau: Y Chapeau residuals.
- Yresidus: Y residuals.

mb.model: List of sPLS models computed for each block.

n.comp: Number of components selected.

n.varX: Number of variables selected for each block.

call: call function

X_input: X input matrix

Y_input: Y input matrix

B.hat: PLS beta matrix

R2: PLS R2

SCR: PLS SCR

SCT: PLS SCT

nzv: Variables removed by remove_near_zero_variance or remove_zero_variance.

nz_coeffvar: Variables removed by coefficient variation near zero.

time: time consumed for running the cox analysis.

nzv: Variables removed by remove_near_zero_variance or remove_zero_variance.

nz_coeffvar: Variables removed by coefficient variation near zero.

time: time consumed for running the cox analysis.

Author(s)

Pedro Salguero Garcia. Maintainer: pedsalga@upv.edu.es

References

Rohart F, Gautier B, Singh A, Cao KAL (2017). "mixOmics: An R package for 'omics feature selection and multiple data integration." *PLoS Computational Biology*, **13**(11). ISSN 15537358, https://pubmed.ncbi.nlm.nih.gov/29099853/.

Examples

```
data("X_multiomic")
data("Y_multiomic")
X <- X_multiomic
X$mirna <- X$mirna[,1:50]
X$proteomic <- X$proteomic[,1:50]
Y <- Y_multiomic
mb.splsdacox(X, Y, n.comp = 2, vector = NULL, x.center = TRUE, x.scale = TRUE)
```

`mb.splsdrcox` *MB.sPLS-DRCOX*

Description

The MB.sPLS-DRCOX function conducts a multi-block sparse partial least squares deviant residuals Cox (MB.sPLS-DRCOX) using a dynamic variable selection approach. This analysis is particularly suited for high-dimensional datasets where the goal is to identify the relationship between explanatory variables and survival outcomes. The function outputs a model of class "Coxmos" with an attribute labeled "MB.sPLS-DRCOX".

Usage

```
mb.splsdrcox(
  X,
  Y,
  n.comp = 4,
  vector = NULL,
  MIN_NVAR = 10,
  MAX_NVAR = 10000,
  n.cut_points = 5,
  EVAL_METHOD = "AUC",
  x.center = TRUE,
  x.scale = FALSE,
  remove_near_zero_variance = TRUE,
  remove_zero_variance = TRUE,
  toKeep.zv = NULL,
  remove_non_significant = TRUE,
  alpha = 0.05,
  MIN_AUC_INCREASE = 0.01,
  pred.method = "cenROC",
  max.iter = 200,
  times = NULL,
  max_time_points = 15,
  MIN_EPV = 5,
  returnData = TRUE,
```

```
  verbose = FALSE
)
```

Arguments

X	Numeric matrix or data.frame. Explanatory variables. Qualitative variables must be transform into binary variables.
Y	Numeric matrix or data.frame. Response variables. Object must have two columns named as "time" and "event". For event column, accepted values are: 0/1 or FALSE/TRUE for censored and event observations.
n.comp	Numeric. Number of latent components to compute for the (s)PLS model (default: 10).
vector	Numeric vector. Used for computing best number of variables. As many values as components have to be provided. If vector = NULL, an automatic detection is perform (default: NULL). If vector is a list, must be named as the names of X param followed by the number of variables to select.
MIN_NVAR	Numeric. Minimum range size for computing cut points to select the best number of variables to use (default: 10).
MAX_NVAR	Numeric. Maximum range size for computing cut points to select the best number of variables to use (default: 1000).
n.cut_points	Numeric. Number of cut points for searching the optimal number of variables. If only two cut points are selected, minimum and maximum size are used. For MB approaches as many as n.cut_points^n.blocks models will be computed as minimum (default: 5).
EVAL_METHOD	Character. If EVAL_METHOD = "AUC", AUC metric will be use to compute the best number of variables. In other case, c-index metric will be used (default: "AUC").
x.center	Logical. If x.center = TRUE, X matrix is centered to zero means (default: TRUE).
x.scale	Logical. If x.scale = TRUE, X matrix is scaled to unit variances (default: FALSE).
remove_near_zero_variance	Logical. If remove_near_zero_variance = TRUE, near zero variance variables will be removed (default: TRUE).
remove_zero_variance	Logical. If remove_zero_variance = TRUE, zero variance variables will be removed (default: TRUE).
toKeep.zv	Character vector. Name of variables in X to not be deleted by (near) zero variance filtering (default: NULL).
remove_non_significant	Logical. If remove_non_significant = TRUE, non-significant variables/components in final cox model will be removed until all variables are significant by forward selection (default: FALSE).
alpha	Numeric. Numerical values are regarded as significant if they fall below the threshold (default: 0.05).

`MIN_AUC_INCREASE`
: Numeric. Minimum improvement between different cross validation models to continue evaluating higher values in the multiple tested parameters. If it is not reached for next 'MIN_COMP_TO_CHECK' models and the minimum 'MIN_AUC' value is reached, the evaluation stops (default: 0.01).

`pred.method`
: Character. AUC evaluation algorithm method for evaluate the model performance. Must be one of the following: "risksetROC", "survivalROC", "cenROC", "nsROC", "smoothROCtime_C", "smoothROCtime_I" (default: "cenROC").

`max.iter`
: Numeric. Maximum number of iterations for PLS convergence (default: 200).

`times`
: Numeric vector. Time points where the AUC will be evaluated. If NULL, a maximum of 'max_time_points' points will be selected equally distributed (default: NULL).

`max_time_points`
: Numeric. Maximum number of time points to use for evaluating the model (default: 15).

`MIN_EPV`
: Numeric. Minimum number of Events Per Variable (EPV) you want reach for the final cox model. Used to restrict the number of variables/components can be computed in final cox models. If the minimum is not meet, the model cannot be computed (default: 5).

`returnData`
: Logical. Return original and normalized X and Y matrices (default: TRUE).

`verbose`
: Logical. If verbose = TRUE, extra messages could be displayed (default: FALSE).

Details

The MB.sPLS-DRCOX methodology is designed to handle multi-block datasets, where each block represents a set of related variables. By employing a sparse partial least squares approach, the function efficiently selects relevant variables from each block, ensuring that the final model is both interpretable and predictive. The Cox proportional hazards model is then applied to the selected variables to assess their association with survival outcomes.

The function offers flexibility in terms of parameter tuning. For instance, users can specify the number of latent components to compute, the range of variables to consider for optimal selection, and the evaluation metric (either AUC or c-index). Additionally, data preprocessing options are available, such as centering and scaling of the explanatory variables, and removal of variables with near-zero or zero variance.

Value

Instance of class "Coxmos" and model "MB.sPLS-DRCOX". The class contains the following elements: `X`: List of normalized X data information.

- `(data)`: normalized X matrix
- `(weightings)`: PLS weights
- `(weightings_norm)`: PLS normalize weights
- `(W.star)`: PLS W* vector
- `(scores)`: PLS scores/variates

- (E): error matrices
- (x.mean): mean values for X matrix
- (x.sd): standard deviation for X matrix

Y: List of normalized Y data information.

- (deviance_residuals): deviance residual vector used as Y matrix in the sPLS.
- (dr.mean): mean values for deviance residuals Y matrix
- (dr.sd): standard deviation for deviance residuals Y matrix'
- (data): normalized X matrix
- (y.mean): mean values for Y matrix
- (y.sd): standard deviation for Y matrix'

survival_model: List of survival model information.

- fit: coxph object.
- AIC: AIC of cox model.
- BIC: BIC of cox model.
- lp: linear predictors for train data.
- coef: Coefficients for cox model.
- YChapeau: Y Chapeau residuals.
- Yresidus: Y residuals.

mb.model: List of sPLS models computed for each block.

n.comp: Number of components selected.

n.varX: Number of variables selected for each block.

call: call function

X_input: X input matrix

Y_input: Y input matrix

B.hat: PLS beta matrix

R2: PLS R2

SCR: PLS SCR

SCT: PLS SCT

nzv: Variables removed by remove_near_zero_variance or remove_zero_variance.

nz_coeffvar: Variables removed by coefficient variation near zero.

time: time consumed for running the cox analysis.

Author(s)

Pedro Salguero Garcia. Maintainer: pedsalga@upv.edu.es

References

Rohart F, Gautier B, Singh A, Cao KAL (2017). "mixOmics: An R package for 'omics feature selection and multiple data integration." *PLoS Computational Biology*, **13**(11). ISSN 15537358, https://pubmed.ncbi.nlm.nih.gov/29099853/.

Examples

```
data("X_multiomic")
data("Y_multiomic")
X <- X_multiomic
X$mirna <- X$mirna[,1:50]
X$proteomic <- X$proteomic[,1:50]
Y <- Y_multiomic
mb.splsdrcox(X, Y, n.comp = 2, vector = NULL, x.center = TRUE, x.scale = TRUE)
```

`norm01` *norm01*

Description

Normalize all values into 0-1 range.

Usage

```
norm01(x)
```

Arguments

x	Numeric matrix or data.frame. Explanatory variables. Only qualitative variables will be transformed into binary variables.

`NR` *The normal reference bandwidth selection for weighted data*

Description

This function computes the data-driven bandwidth for smoothing the ROC (or distribution) function using the NR method of Beyene and El Ghouch (2020). This is an extension of the classical (unweighted) normal reference bandwith selection method to the case of weighted data.

Usage

```
NR(X, wt, ktype = "normal")
```

Arguments

X	The numeric data vector.
wt	The non-negative weight vector.
ktype	A character string giving the type kernel to be used: "normal", "epanechnikov", "biweight", or "triweight". By default, the "normal" kernel is used.

Details

See Beyene and El Ghouch (2020) for details.

Value

Returns the computed value for the bandwith parameter.

Author(s)

Kassu Mehari Beyene, Catholic University of Louvain. <kasu.beyene@uclouvain.be>

Anouar El Ghouch, Catholic University of Louvain. <anouar.elghouch@uclouvain.be>

References

Beyene, K. M. and El Ghouch A. (2020). Smoothed time-dependent ROC curves for right-censored survival data. *submitted*.

PI *The plug-in bandwidth selection for weighted data*

Description

This function computes the data-driven bandwidth for smoothing the ROC (or distribution) function using the PI method of Beyene and El Ghouch (2020). This is an extension of the classical (unweighted) direct plug-in bandwith selection method to the case of weighted data.

Usage

```
PI(X, wt, ktype = "normal")
```

Arguments

X	The numeric vector of random variable.
wt	The non-negative weight vector.
ktype	A character string giving the type kernel to be used: "normal", "epanechnikov", "biweight", or "triweight". By default, the "normal" kernel is used.

Details

See Beyene and El Ghouch (2020) for details.

Value

Returns the computed value for the bandwith parameter.

Author(s)

Kassu Mehari Beyene, Catholic University of Louvain. <kasu.beyene@uclouvain.be>

Anouar El Ghouch, Catholic University of Louvain. <anouar.elghouch@uclouvain.be>

References

Beyene, K. M. and El Ghouch A. (2020). Smoothed time-dependent ROC curves for right-censored survival data. *submitted*.

`plot_cox.event` *plot_cox.event*

Description

Visualizes the distribution of events based on a Coxmos model's predictions. The function provides both density and histogram plots to elucidate the event distribution, which can be instrumental in understanding the model's behavior across different prediction types.

Usage

```
plot_cox.event(model, type = "lp", n.breaks = 20)
```

Arguments

`model`	Coxmos model.
`type`	Character. Prediction type: "lp", "risk", "expected" or "survival" (default: "lp").
`n.breaks`	Numeric. If BREAKTIME is NULL, "n.breaks" is the number of time-break points to compute (default: 20).

Details

The function takes in a Coxmos model and, based on the specified prediction type (`lp`, `risk`, `expected`, or `survival`), computes the respective predictions. The `lp` (linear predictor) is the default prediction type. The density and histogram plots are then generated to represent the distribution of events (censored or occurred) concerning these predictions.

The density plot provides a smoothed representation of the event distribution, with separate curves for censored and occurred events. This visualization can be particularly useful to discern the overall distribution and overlap between the two event types.

The histogram, on the other hand, offers a binned representation of the event distribution. Each bin's height represents the count of observations falling within that prediction range, stacked by event type. This visualization provides a more granular view of the event distribution across different prediction values.

It's imperative to note that the models should be run with the `returnData = TRUE` option to ensure the necessary data is available for plotting.

Value

A list containing three elements: `df`: A data.frame with the computed predictions based on the specified type and the corresponding event status. `plot.density`: A ggplot object representing the density plot of the event distribution, with separate curves for censored and occurred events. `plot.histogram`: A ggplot object representing the histogram of the event distribution, with bins stacked by event type.

Author(s)

Pedro Salguero Garcia. Maintainer: pedsalga@upv.edu.es

Examples

```
data("X_proteomic")
data("Y_proteomic")
X <- X_proteomic[,1:50]
Y <- Y_proteomic
splsicox.model <- splsicox(X, Y, n.comp = 2, penalty = 0.5, x.center = TRUE, x.scale = TRUE)
plot_cox.event(splsicox.model)
```

---

plot_cox.event.list *plot_cox.event.list*

Description

Run the function "plot_cox.event" for a list of models. More information in "?plot_cox.event".

Usage

```
plot_cox.event.list(lst_models, type = "lp", n.breaks = 20)
```

Arguments

`lst_models`	List of Coxmos models.
`type`	Character. Prediction type: "lp", "risk", "expected" or "survival" (default: "lp").
`n.breaks`	Numeric. Number of time-break points to compute (default: 20).

Value

A list containing three elements per each model: `df`: A data.frame with the computed predictions based on the specified type and the corresponding event status. `plot.density`: A ggplot object representing the density plot of the event distribution, with separate curves for censored and occurred events. `plot.histogram`: A ggplot object representing the histogram of the event distribution, with bins stacked by event type.

Author(s)

Pedro Salguero Garcia. Maintainer: pedsalga@upv.edu.es

Examples

```
data("X_proteomic")
data("Y_proteomic")
X <- X_proteomic[,1:50]
Y <- Y_proteomic
splsicox.model <- splsicox(X, Y, n.comp = 2, penalty = 0.5, x.center = TRUE, x.scale = TRUE)
splsdrcox.model <- splsdrcox(X, Y, n.comp = 2, penalty = 0.5, x.center = TRUE, x.scale = TRUE)
lst_models = list("sPLSICOX" = splsicox.model, "sPLSDRCOX" = splsdrcox.model)
plot_cox.event.list(lst_models)
```

---

plot_Coxmos.MB.PLS.model

plot_Coxmos.MB.PLS.model

Description

Visualizes the Coxmos model using multiblock partial least squares (MB-PLS) approach. This function offers various plotting modes, including scores, loadings, and biplot visualizations, to provide insights into the model's structure and relationships.

Usage

```
plot_Coxmos.MB.PLS.model(
  model,
  comp = c(1, 2),
  mode = "scores",
  factor = NULL,
  legend_title = NULL,
  top = NULL,
  only_top = FALSE,
  radius = NULL,
  names = TRUE,
  colorReverse = FALSE,
  text.size = 2,
  overlaps = 10
)
```

Arguments

model	Coxmos model.
comp	Numeric vector. Vector of length two. Select which components to plot (default: c(1,2)).

mode	Character. Choose one of the following plots: "scores", "loadings" o "biplot" (default: "scores").
factor	Factor. Factor variable to color the observations. If factor = NULL, event will be used (default: NULL).
legend_title	Character. Legend title (default: NULL).
top	Numeric. Show "top" first variables. If top = NULL, all variables are shown (default: NULL).
only_top	Logical. If "only_top" = TRUE, then only top/radius loading variables will be shown in loading or biplot graph (default: FALSE).
radius	Numeric. Radius size (loading/scale value) to plot variable names that are greater than the radius value (default: NULL).
names	Logical. Show loading names for top variables or for those that are outside the radius size (default: TRUE).
colorReverse	Logical. Reverse palette colors (default: FALSE).
text.size	Numeric. Text size (default: 2).
overlaps	Numeric. Number of overlaps to show when plotting loading names (default: 10).

Details

The plot_Coxmos.MB.PLS.model function is designed to generate comprehensive visualizations of the Coxmos model, specifically tailored for multiblock PLS. It leverages the inherent structure of the model to produce plots that can aid in the interpretation of the model's components and their relationships.

Depending on the chosen mode, the function can display:

- Scores: This mode visualizes the scores of the model, which represent the projections of the original data onto the PLS components. The scores can be colored by a factor variable, and ellipses can be added to represent the distribution of the scores.
- Loadings: This mode displays the loadings of the model, which indicate the contribution of each variable to the PLS components. The loadings can be filtered by a specified threshold (top or radius), and arrows can be added to represent the direction and magnitude of the loadings.
- Biplot: A biplot combines both scores and loadings in a single plot, providing a comprehensive view of the relationships between the observations and variables in the model.

The function also offers various customization options, such as adjusting the text size, reversing the color palette, and specifying the number of overlaps for loading names. It ensures that the visualizations are informative and tailored to the user's preferences and the specific characteristics of the data.

It's important to note that the function performs checks to ensure the input model is of the correct class and provides informative messages for any inconsistencies detected.

plot_Coxmos.PLS.model *plot_Coxmos.PLS.model*

Description

Visualizes the Coxmos model using partial least squares (PLS) approach. This function offers various plotting modes, including scores, loadings, and biplot visualizations, to provide insights into the model's structure and relationships.

Usage

```
plot_Coxmos.PLS.model(
  model,
  comp = c(1, 2),
  mode = "scores",
  factor = NULL,
  legend_title = NULL,
  top = NULL,
  only_top = FALSE,
  radius = NULL,
  names = TRUE,
  colorReverse = FALSE,
  text.size = 2,
  overlaps = 10
)
```

Arguments

model	Coxmos model.
comp	Numeric vector. Vector of length two. Select which components to plot (default: c(1,2)).
mode	Character. Choose one of the following plots: "scores", "loadings" o "biplot" (default: "scores").
factor	Factor. Factor variable to color the observations. If factor = NULL, event will be used (default: NULL).
legend_title	Character. Legend title (default: NULL).
top	Numeric. Show "top" first variables. If top = NULL, all variables are shown (default: NULL).
only_top	Logical. If "only_top" = TRUE, then only top/radius loading variables will be shown in loading or biplot graph (default: FALSE).
radius	Numeric. Radius size (loading/scale value) to plot variable names that are greater than the radius value (default: NULL).
names	Logical. Show loading names for top variables or for those that are outside the radius size (default: TRUE).

colorReverse	Logical. Reverse palette colors (default: FALSE).
text.size	Numeric. Text size (default: 2).
overlaps	Numeric. Number of overlaps to show when plotting loading names (default: 10).

Details

The plot_Coxmos.PLS.model function is designed to generate comprehensive visualizations of the Coxmos model, specifically tailored for PLS. It leverages the inherent structure of the model to produce plots that can aid in the interpretation of the model's components and their relationships.

Depending on the chosen mode, the function can display:

- Scores: This mode visualizes the scores of the model, which represent the projections of the original data onto the PLS components. The scores can be colored by a factor variable, and ellipses can be added to represent the distribution of the scores.
- Loadings: This mode displays the loadings of the model, which indicate the contribution of each variable to the PLS components. The loadings can be filtered by a specified threshold (top or radius), and arrows can be added to represent the direction and magnitude of the loadings.
- Biplot: A biplot combines both scores and loadings in a single plot, providing a comprehensive view of the relationships between the observations and variables in the model.

The function also offers various customization options, such as adjusting the text size, reversing the color palette, and specifying the number of overlaps for loading names. It ensures that the visualizations are informative and tailored to the user's preferences and the specific characteristics of the data.

It's important to note that the function performs checks to ensure the input model is of the correct class and provides informative messages for any inconsistencies detected.

---

plot_divergent.biplot *plot_divergent.biplot*

Description

Generates a divergent biplot visualizing the distribution of a qualitative variable against a quantitative variable, further categorized by an event matrix.

Usage

```
plot_divergent.biplot(
  X,
  Y,
  NAMEVAR1,
  NAMEVAR2,
  BREAKTIME,
  x.text = "N. of Samples"
)
```

Arguments

X	Numeric matrix or data.frame. Explanatory variables with "NAMEVAR1" and "NAMEVAR2" variables. "NAMEVAR1" must be a factor variable.
Y	Numeric matrix or data.frame. Response variables. Object must have two columns named as "time" and "event". For event column, accepted values are: 0/1 or FALSE/TRUE for censored and event observations.
NAMEVAR1	Character. Factor variable name (must be located in colnames(X) and have to have two levels).
NAMEVAR2	Character. Numerical variable name (must be located in colnames(X)).
BREAKTIME	Numeric. Size of time to split the data into "total_time / BREAKTIME + 1" points. If BREAKTIME = NULL, "n.breaks" is used (default: NULL).
x.text	Character. Title for X axis.

Details

The function `plot_divergent.biplot` is designed to offer a comprehensive visualization of the relationship between a qualitative and a quantitative variable, while also taking into account an associated event matrix. The qualitative variable, denoted by "NAMEVAR1", is expected to be a factor with two levels, and the quantitative variable, "NAMEVAR2", is numerically represented. The event matrix, "Y", consists of two columns: "time" and "event". The "event" column indicates whether an observation is censored or an event, represented by binary values (0/1 or FALSE/TRUE).

The function processes the input data to categorize the quantitative variable into groups based on the specified "BREAKTIME" parameter. Each group represents a range of values for the quantitative variable. The resulting plot displays the number of samples for each level of the qualitative variable on the X-axis, while the Y-axis represents the categorized groups of the quantitative variable. The bars in the plot are further colored based on the event type, providing a clear distinction between censored and event observations.

Value

A 'ggplot2' two side bar plot. X axis represent the number of samples per each NAMEVAR1 factor levels and Y axis, the X NAMEVAR2 numerical variables categorize in groups of breaks.

Author(s)

Pedro Salguero Garcia. Maintainer: pedsalga@upv.edu.es

Examples

```
X <- data.frame(sex = factor(c("M","M","F","F","F","M","F","M","M")),
age = as.numeric(c(22,23,25,28,32,30,29,33,32)))

Y = data.frame(time = c(24,25,28,29,22,26,22,23,24),
event = c(TRUE,TRUE,FALSE,TRUE,FALSE,TRUE,TRUE,FALSE,FALSE))

NAMEVAR1 = "sex"
NAMEVAR2 = "age"
plot_divergent.biplot(X, Y, NAMEVAR1, NAMEVAR2, BREAKTIME = 5, x.text = "N. of Patients")
```

plot_evaluation *plot_evaluation*

Description

Generates a comprehensive evaluation of the performance of a given Coxmos evaluation object from `eval_Coxmos_models()`, offering both statistical tests and visual plots for assessment.

Usage

```
plot_evaluation(
  eval_results,
  evaluation = "AUC",
  pred.attr = "mean",
  y.min = NULL,
  type = "both",
  round_times = FALSE,
  decimals = 2,
  title = NULL,
  title_size_text = 15,
  legend_title = "Method",
  legend_size_text = 12,
  x_axis_size_text = 10,
  y_axis_size_text = 10,
  label_x_axis_size = 10,
  label_y_axis_size = 10
)
```

Arguments

`eval_results`	Coxmos evaluation object from `eval_Coxmos_models()`.
`evaluation`	Character. Perform the evaluation using the "AUC" or "Brier" metric (default: "AUC").
`pred.attr`	Character. Way to evaluate the metric selected. Must be one of the following: "mean" or "median" (default: "mean").
`y.min`	Numeric. Minimum Y value for establish the Y axis value. If y.min = NULL, automatic detection is performed (default: NULL).
`type`	Character. Plot type. Must be one of the following: "both", "line" or "mean". In other case, "both" will be selected (default: "both").
`round_times`	Logical. Whether times x value should be rounded (default: FALSE).
`decimals`	Numeric. Number of decimals to use in round times. Must be a value greater or equal zero (default = 2).
`title`	Character. Plot title (default: NULL).
`title_size_text`	Numeric. Text size for legend title (default: 15).

`legend_title` Character. Legend title (default: "Method").

`legend_size_text`
Numeric. Text size for legend title (default: 12).

`x_axis_size_text`
Numeric. Text size for x axis (default: 10).

`y_axis_size_text`
Numeric. Text size for y axis (default: 10).

`label_x_axis_size`
Numeric. Text size for x label axis (default: 10).

`label_y_axis_size`
Numeric. Text size for y label axis (default: 10).

Details

The `plot_evaluation` function is designed to facilitate a rigorous evaluation of the performance of models, specifically in the context of survival analysis. This function is tailored to work with a Coxmos evaluation object, which encapsulates the results of survival models. The primary objective is to provide both statistical and visual insights into the model's performance.

The function offers flexibility in the evaluation metric, allowing users to choose between the Area Under the Curve (AUC) and the Brier score. The chosen metric is then evaluated based on either its mean or median value, as specified by the "pred.attr" parameter. The resulting plots can be tailored to display continuous performance over time or aggregated mean performance, based on the "type" parameter.

A salient feature of this function is its ability to conduct statistical tests to compare the performance across different methods. Supported tests include the t-test, ANOVA, Wilcoxon rank-sum test, and Kruskal-Wallis test. These tests provide a quantitative measure of the differences in performance, aiding in the objective assessment of the models.

The visual outputs are generated using the 'ggplot2' package, ensuring high-quality and interpretable plots. The function also offers extensive customization options for the plots, including axis labels, title, and text sizes, ensuring that the outputs align with the user's preferences and the intended audience's expectations.

Value

A list of lst_eval_results length. Each element is a list of three elements. `lst_plots`: A list of two plots. The evaluation over the time, and the extension adding the mean or median on the right. `lst_plot_comparisons`: A list of comparative boxplots by t.test, anova, wilcoxon, kruscal. `df`: Data.frame of evaluation result.

Author(s)

Pedro Salguero Garcia. Maintainer: pedsalga@upv.edu.es

Examples

```
data("X_proteomic")
data("Y_proteomic")
set.seed(123)
```

```
index_train <- caret::createDataPartition(Y_proteomic$event, p = .5, list = FALSE, times = 1)
X_train <- X_proteomic[index_train,1:50]
Y_train <- Y_proteomic[index_train,]
X_test <- X_proteomic[-index_train,1:50]
Y_test <- Y_proteomic[-index_train,]
coxEN.model <- coxEN(X_train, Y_train, x.center = TRUE, x.scale = TRUE)
eval_results <- eval_Coxmos_models(lst_models = list("coxEN" = coxEN.model), X_test = X_test,
Y_test = Y_test)
plot_eval_results <- plot_evaluation(eval_results)
```

`plot_evaluation.list` *plot_evaluation.list*

Description

Run the function "plot_evaluation" for a list of results. More information in "?plot_evaluation".

Usage

```
plot_evaluation.list(
  lst_eval_results,
  evaluation = "AUC",
  pred.attr = "mean",
  y.min = NULL,
  type = "both",
  round_times = FALSE,
  decimals = 2,
  title = NULL,
  title_size_text = 15,
  legend_title = "Method",
  legend_size_text = 12,
  x_axis_size_text = 10,
  y_axis_size_text = 10,
  label_x_axis_size = 10,
  label_y_axis_size = 10
)
```

Arguments

`lst_eval_results`	List (named) of Coxmos evaluation results from `eval_Coxmos_models()`.
`evaluation`	Character. Perform the evaluation using the "AUC" or "Brier" metric (default: "AUC").
`pred.attr`	Character. Way to evaluate the metric selected. Must be one of the following: "mean" or "median" (default: "mean").
`y.min`	Numeric. Minimum Y value for establish the Y axis value. If y.min = NULL, automatic detection is performed (default: NULL).

type	Character. Plot type. Must be one of the following: "both", "line" or "mean". In other case, "both" will be selected (default: "both").
round_times	Logical. Whether times x value should be rounded (default: FALSE).
decimals	Numeric. Number of decimals to use in round times. Must be a value greater or equal zero (default = 2).
title	Character. Plot title (default: NULL).
title_size_text	Numeric. Text size for legend title (default: 15).
legend_title	Character. Legend title (default: "Method").
legend_size_text	Numeric. Text size for legend title (default: 12).
x_axis_size_text	Numeric. Text size for x axis (default: 10).
y_axis_size_text	Numeric. Text size for y axis (default: 10).
label_x_axis_size	Numeric. Text size for x label axis (default: 10).
label_y_axis_size	Numeric. Text size for y label axis (default: 10).

Value

A list of lst_eval_results length. Each element is a list of three elements. `lst_plots`: A list of two plots. The evaluation over the time, and the extension adding the mean or median on the right. `lst_plot_comparisons`: A list of comparative boxplots by t.test, anova, wilcoxon, kruscal. `df`: Data.frame of evaluation result.

Author(s)

Pedro Salguero Garcia. Maintainer: pedsalga@upv.edu.es

Examples

```
data("X_proteomic")
data("Y_proteomic")
set.seed(123)
index_train <- caret::createDataPartition(Y_proteomic$event, p = .5, list = FALSE, times = 1)
X_train <- X_proteomic[index_train,1:50]
Y_train <- Y_proteomic[index_train,]
X_test <- X_proteomic[-index_train,1:50]
Y_test <- Y_proteomic[-index_train,]
coxEN.model <- coxEN(X_train, Y_train, x.center = TRUE, x.scale = TRUE)
eval_results <- list()
eval_results[["cenROC"]] <- eval_Coxmos_models(lst_models = list("coxEN" = coxEN.model),
X_test = X_test, Y_test = Y_test, pred.method = "cenROC")
eval_results[["survivalROC"]] <- eval_Coxmos_models(lst_models = list("coxEN" = coxEN.model),
X_test = X_test, Y_test = Y_test, pred.method = "survivalROC")
plot_eval_results <- plot_evaluation.list(eval_results)
```

plot_events *plot_events*

Description

Generates a bar plot visualizing the distribution of events over time, categorizing observations as either censored or non-censored.

Usage

```
plot_events(
  Y,
  max.breaks = 20,
  roundTo = 0.1,
  categories = c("Censored", "Death"),
  y.text = "Number of observations",
  verbose = FALSE
)
```

Arguments

Y	Numeric matrix or data.frame. Response variables. Object must have two columns named as "time" and "event". For event column, accepted values are: 0/1 or FALSE/TRUE for censored and event observations.
max.breaks	Numeric. Maximum number of breaks in X axis (default: 20).
roundTo	Numeric. Value to round time. If roundTo = 0.1, the results will be rounded to the tenths (default: 0.1).
categories	Character vector. Vector of length two to name both categories for censored and non-censored observations (default: c("Censored","Death")).
y.text	Character. Y axis title (default: "Number of observations").
verbose	Logical. If verbose = TRUE, extra messages could be displayed (default: FALSE).

Details

The plot_events function is meticulously crafted to provide a visualization of event occurrences over a specified time frame. The primary objective of this function is to elucidate the distribution of events, distinguishing between censored and non-censored observations. The input response matrix, "Y", is expected to encompass two pivotal columns: "time" and "event". The "time" column delineates the temporal occurrence of each observation, while the "event" column demarcates whether an observation is censored or an event, with accepted binary representations being 0/1 or FALSE/TRUE.

The function employs a systematic approach to categorize the time variable into distinct intervals or "breaks". The number of these intervals is determined by the "max.breaks" parameter, and their size is influenced by the "roundTo" parameter. Each interval represents a range of time values, and the resulting plot showcases the number of censored and non-censored observations within each interval. The bars in the plot are color-coded based on the event type, offering a clear visual distinction between the two categories.

Value

A list of two elements. `plot`: Barplot. `df`: Data.frame used for the plotting.

Author(s)

Pedro Salguero Garcia. Maintainer: pedsalga@upv.edu.es

Examples

```
data("X_proteomic")
data("Y_proteomic")
Y_train <- Y_proteomic
plot_events(Y_train, categories = c("Censored","Death"))
```

`plot_forest` *plot_forest*

Description

Generates a forest plot for Coxmos models, visualizing the hazard ratios and their confidence intervals. The function leverages the capabilities of the `survminer::ggforest` function to produce a comprehensive representation of the model's coefficients.

Usage

```
plot_forest(
  model,
  title = "Hazard Ratio",
  cpositions = c(0.02, 0.22, 0.4),
  fontsize = 0.7,
  refLabel = "reference",
  noDigits = 2
)
```

Arguments

`model`	Coxmos model.
`title`	Character. Forest plot title (default: "Hazard Ratio").
`cpositions`	Numeric vector. Relative positions of first three columns in the OX scale (default: c(0.02, 0.22, 0.4)).
`fontsize`	Numeric. Elative size of annotations in the plot (default: 0.7).
`refLabel`	Character. Label for reference levels of factor variables (default: "reference").
`noDigits`	Numeric. Number of digits for estimates and p-values in the plot (default: 2).

Details

The forest plot is a graphical representation of the point estimates and confidence intervals of the hazard ratios derived from a Coxmos model. Each row in the plot corresponds to a variable or component from the model, with a point representing the hazard ratio and horizontal lines indicating the confidence intervals. The plot provides a visual assessment of the significance and magnitude of each variable's effect on the outcome.

The function starts by validating the provided model to ensure it belongs to the Coxmos class and is among the recognized Coxmos models. If the model is valid, the function then proceeds to generate the forest plot using the `survminer::ggforest` function. Several customization options are available, including adjusting the title, column positions, font size, reference label, and the number of digits displayed for estimates and p-values.

Forest plots are instrumental in the field of survival analysis, offering a concise visualization of the model's results, making them easier to interpret and communicate.

Value

A ggplot object representing the forest plot. The plot visualizes the hazard ratios and their confidence intervals for each variable or component from the Coxmos model.

Author(s)

Pedro Salguero Garcia. Maintainer: pedsalga@upv.edu.es

Examples

```
data("X_proteomic")
data("Y_proteomic")
X <- X_proteomic[,1:50]
Y <- Y_proteomic
splsicox.model <- splsicox(X, Y, n.comp = 2, penalty = 0.5, x.center = TRUE, x.scale = TRUE)
plot_forest(splsicox.model)
```

`plot_forest.list` *plot_forest.list*

Description

Run the function "plot_forest" for a list of models. More information in "?plot_forest".

Usage

```
plot_forest.list(
  lst_models,
  title = "Hazard Ratio",
  cpositions = c(0.02, 0.22, 0.4),
  fontsize = 0.7,
  refLabel = "reference",
```

```
  noDigits = 2
)
```

Arguments

lst_models	List of Coxmos models.
title	Character. Forest plot title (default: "Hazard Ratio").
cpositions	Numeric vector. Relative positions of first three columns in the OX scale (default: c(0.02, 0.22, 0.4)).
fontsize	Numeric. Elative size of annotations in the plot (default: 0.7).
refLabel	Character. Label for reference levels of factor variables (default: "reference").
noDigits	Numeric. Number of digits for estimates and p-values in the plot (default: 2).

Value

A ggplot object per model representing the forest plot. The plot visualizes the hazard ratios and their confidence intervals for each variable or component from the Coxmos model.

Author(s)

Pedro Salguero Garcia. Maintainer: pedsalga@upv.edu.es

Examples

```
data("X_proteomic")
data("Y_proteomic")
X <- X_proteomic[,1:50]
Y <- Y_proteomic
splsicox.model <- splsicox(X, Y, n.comp = 2, penalty = 0.5, x.center = TRUE, x.scale = TRUE)
splsdrcox.model <- splsdrcox(X, Y, n.comp = 2, penalty = 0.5, x.center = TRUE, x.scale = TRUE)
lst_models = list("sPLSICOX" = splsicox.model, "sPLSDRCOX" = splsdrcox.model)
plot_forest.list(lst_models)
```

---

plot_LP.multipleObservations

plot_LP.multipleObservations

Description

Visualizes the linear predictors for multiple patients based on a given Coxmos model.

Usage

```
plot_LP.multipleObservations(
  model,
  new_observations,
  error.bar = FALSE,
  onlySig = TRUE,
  alpha = 0.05,
  zero.rm = TRUE,
  auto.limits = TRUE,
  top = NULL
)
```

Arguments

model	Coxmos model.
new_observations	Numeric matrix or data.frame. New explanatory variables (raw data). Qualitative variables must be transform into binary variables.
error.bar	Logical. Show error bar (default: FALSE).
onlySig	Logical. Compute plot using only significant components (default: TRUE).
alpha	Numeric. Numerical values are regarded as significant if they fall below the threshold (default: 0.05).
zero.rm	Logical. Remove variables equal to 0 (default: TRUE).
auto.limits	Logical. If "auto.limits" = TRUE, limits are detected automatically (default: TRUE).
top	Numeric. Show "top" first variables. If top = NULL, all variables are shown (default: NULL).

Details

The function `plot_LP.multipleObservations` is designed to visualize the linear predictors for multiple patients based on the provided Coxmos model. The function takes into account various parameters to customize the visualization, such as the significance level, error bars, and the number of top variables to display.

The function works by first checking the class of the provided model. Depending on the model type, it delegates the plotting task to one of the three methods: classical models, PLS models, or multiblock PLS models. Each of these methods is tailored to handle specific model types and produce the desired plots.

Value

A ggplot object visualizing the linear predictors for multiple patients based on the provided Coxmos model.

Author(s)

Pedro Salguero Garcia. Maintainer: pedsalga@upv.edu.es

Examples

```
data("X_proteomic")
data("Y_proteomic")
set.seed(123)
index_train <- caret::createDataPartition(Y_proteomic$event, p = .5, list = FALSE, times = 1)
X_train <- X_proteomic[index_train,1:50]
Y_train <- Y_proteomic[index_train,]
X_test <- X_proteomic[-index_train,1:50]
Y_test <- Y_proteomic[-index_train,]
splsicox.model <- splsicox(X_train, Y_train, n.comp = 2, penalty = 0.5, x.center = TRUE,
x.scale = TRUE)
plot_LP.multipleObservations(model = splsicox.model, new_observations = X_test[1:5,])
```

---

plot_LP.multipleObservations.list

plot_LP.multipleObservations.list

Description

Run the function "plot_LP.multipleObservations" for a list of models. More information in "?plot_LP.multipleObservations".

Usage

```
plot_LP.multipleObservations.list(
  lst_models,
  new_observations,
  error.bar = FALSE,
  onlySig = TRUE,
  alpha = 0.05,
  zero.rm = TRUE,
  auto.limits = TRUE,
  top = NULL
)
```

Arguments

lst_models	List of Coxmos models.
new_observations	Numeric matrix or data.frame. New explanatory variables (raw data). Qualitative variables must be transform into binary variables.
error.bar	Logical. Show error bar (default: FALSE).
onlySig	Logical. Compute plot using only significant components (default: TRUE).
alpha	Numeric. Numerical values are regarded as significant if they fall below the threshold (default: 0.05).
zero.rm	Logical. Remove variables equal to 0 (default: TRUE).

auto.limits	Logical. If "auto.limits" = TRUE, limits are detected automatically (default: TRUE).
top	Numeric. Show "top" first variables. If top = NULL, all variables are shown (default: NULL).

Value

A list of ggplot objects for each model in the lst_models. Each plot visualizes the linear predictor values for multiple patients based on the specified Coxmos model. The plots can optionally display error bars, consider only significant components, and can be limited to a specified number of top variables. The visualization aids in understanding the influence of explanatory variables on the survival prediction for each patient in the context of the provided models.

Author(s)

Pedro Salguero Garcia. Maintainer: pedsalga@upv.edu.es

Examples

```
data("X_proteomic")
data("Y_proteomic")
set.seed(123)
index_train <- caret::createDataPartition(Y_proteomic$event, p = .5, list = FALSE, times = 1)
X_train <- X_proteomic[index_train,1:50]
Y_train <- Y_proteomic[index_train,]
X_test <- X_proteomic[-index_train,1:50]
Y_test <- Y_proteomic[-index_train,]
splsicox.model <- splsicox(X_train, Y_train, n.comp = 2, penalty = 0.5, x.center = TRUE,
x.scale = TRUE)
splsdrcox.model <- splsdrcox(X_train, Y_train, n.comp = 2, penalty = 0.5, x.center = TRUE,
x.scale = TRUE)
lst_models = list("sPLSICOX" = splsicox.model, "sPLSDRCOX" = splsdrcox.model)
plot_LP.multipleObservations.list(lst_models = lst_models, X_test[1:5,])
```

---

plot_observation.eventDensity

plot_observation.eventDensity

Description

Visualizes the event density for a given observation's data using the Coxmos model.

Usage

```
plot_observation.eventDensity(
  observation,
  model,
  time = NULL,
```

```
  type = "lp",
  size = 3,
  color = "red"
)
```

Arguments

observation	Numeric matrix or data.frame. New explanatory variables (raw data) for one observation. Qualitative variables must be transform into binary variables.
model	Coxmos model.
time	Numeric. Time point where the AUC will be evaluated (default: NULL).
type	Character. Prediction type: "lp", "risk", "expected" or "survival" (default: "lp").
size	Numeric. Point size (default: 3).
color	String. R Color.

Details

The plot_observation.eventDensity function provides a graphical representation of the event density for a specific observation's data, based on the Coxmos model. The function computes the density of events and non-events and plots them, highlighting the predicted value for the given observation's data. The density is determined using density estimation, and the predicted value is obtained from the Coxmos model. The function allows customization of the plot aesthetics, such as point size and color. The resulting plot provides a visual comparison of the observation's predicted event density against the overall event density distribution, aiding in the interpretation of the observation's risk profile.

Value

A ggplot object representing a density of the predicted event values based on the provided Coxmos model.

Author(s)

Pedro Salguero Garcia. Maintainer: pedsalga@upv.edu.es

Examples

```
data("X_proteomic")
data("Y_proteomic")
set.seed(123)
index_train <- caret::createDataPartition(Y_proteomic$event, p = .5, list = FALSE, times = 1)
X_train <- X_proteomic[index_train,1:50]
Y_train <- Y_proteomic[index_train,]
X_test <- X_proteomic[-index_train,1:50]
Y_test <- Y_proteomic[-index_train,]
coxEN.model <- coxEN(X_train, Y_train, x.center = TRUE, x.scale = TRUE)
observation = X_test[1,,drop=FALSE]
plot_observation.eventDensity(observation = observation, model = coxEN.model, time = NULL)
```

plot_observation.eventHistogram
plot_observation.eventHistogram

Description

Generates a histogram plot for observation event data based on a given Coxmos model. The function visualizes the distribution of predicted values and highlights the prediction for a specific observation.

Usage

```
plot_observation.eventHistogram(
  observation,
  model,
  time = NULL,
  type = "lp",
  size = 3,
  color = "red"
)
```

Arguments

observation	Numeric matrix or data.frame. New explanatory variables (raw data) for one observation. Qualitative variables must be transform into binary variables.
model	Coxmos model.
time	Numeric. Time point where the AUC will be evaluated (default: NULL).
type	Character. Prediction type: "lp", "risk", "expected" or "survival" (default: "lp").
size	Numeric. Point size (default: 3).
color	String. R Color.

Details

The plot_observation.eventHistogram function is designed to provide a visual representation of the distribution of predicted event values based on a Coxmos model. The function takes in observation data, a specified time point, and a Coxmos model to compute the prediction. The resulting histogram plot displays the distribution of these predictions, with a specific emphasis on the prediction for the provided observation data. The prediction is represented as a point on the histogram, allowing for easy comparison between the specific observation's prediction and the overall distribution of predictions. The type of prediction ("lp", "risk", "expected", or "survival") can be specified, offering flexibility in the kind of insights one wishes to derive from the visualization. The appearance of the point representing the observation's prediction can be customized using the size and color parameters.

Value

A ggplot object representing a histogram of the predicted event values based on the provided Coxmos model.

Author(s)

Pedro Salguero Garcia. Maintainer: pedsalga@upv.edu.es

Examples

```
data("X_proteomic")
data("Y_proteomic")
set.seed(123)
index_train <- caret::createDataPartition(Y_proteomic$event, p = .5, list = FALSE, times = 1)
X_train <- X_proteomic[index_train,1:50]
Y_train <- Y_proteomic[index_train,]
X_test <- X_proteomic[-index_train,1:50]
Y_test <- Y_proteomic[-index_train,]
coxEN.model <- coxEN(X_train, Y_train, x.center = TRUE, x.scale = TRUE)
observation = X_test[1,,drop=FALSE]
plot_observation.eventHistogram(observation = observation, model = coxEN.model, time = NULL)
```

---

plot_PLS_Coxmos *plot_PLS_Coxmos*

Description

Visualizes the Coxmos models based on partial least squares (PLS) or Multi-block PLS approaches. This function offers various plotting modes, including scores, loadings, and biplot visualizations, to provide insights into the model's structure and relationships.

Usage

```
plot_PLS_Coxmos(
  model,
  comp = c(1, 2),
  mode = "scores",
  factor = NULL,
  legend_title = NULL,
  top = NULL,
  only_top = FALSE,
  radius = NULL,
  names = TRUE,
  colorReverse = FALSE,
  text.size = 2,
  overlaps = 10
)
```

Arguments

model	Coxmos model.
comp	Numeric vector. Vector of length two. Select which components to plot (default: c(1,2)).
mode	Character. Choose one of the following plots: "scores", "loadings" o "biplot" (default: "scores").
factor	Factor. Factor variable to color the observations. If factor = NULL, event will be used (default: NULL).
legend_title	Character. Legend title (default: NULL).
top	Numeric. Show "top" first variables. If top = NULL, all variables are shown (default: NULL).
only_top	Logical. If "only_top" = TRUE, then only top/radius loading variables will be shown in loading or biplot graph (default: FALSE).
radius	Numeric. Radius size (loading/scale value) to plot variable names that are greater than the radius value (default: NULL).
names	Logical. Show loading names for top variables or for those that are outside the radius size (default: TRUE).
colorReverse	Logical. Reverse palette colors (default: FALSE).
text.size	Numeric. Text size (default: 2).
overlaps	Numeric. Number of overlaps to show when plotting loading names (default: 10).

Details

The plot_Coxmos.PLS.model function is designed to generate comprehensive visualizations of the Coxmos models. It leverages the inherent structure of the model to produce plots that can aid in the interpretation of the model's components and their relationships.

Depending on the chosen mode, the function can display:

- Scores: This mode visualizes the scores of the model, which represent the projections of the original data onto the PLS components. The scores can be colored by a factor variable, and ellipses can be added to represent the distribution of the scores.
- Loadings: This mode displays the loadings of the model, which indicate the contribution of each variable to the PLS components. The loadings can be filtered by a specified threshold (top or radius), and arrows can be added to represent the direction and magnitude of the loadings.
- Biplot: A biplot combines both scores and loadings in a single plot, providing a comprehensive view of the relationships between the observations and variables in the model.

The function also offers various customization options, such as adjusting the text size, reversing the color palette, and specifying the number of overlaps for loading names. It ensures that the visualizations are informative and tailored to the user's preferences and the specific characteristics of the data.

It's important to note that the function performs checks to ensure the input model is of the correct class and provides informative messages for any inconsistencies detected.

Value

A list of two elements. `plot`: Score, Loading or Biplot graph in 'ggplot2' format. `outliers`: Data.frame of outliers detected in the plot.

Author(s)

Pedro Salguero Garcia. Maintainer: pedsalga@upv.edu.es

Examples

```
data("X_proteomic")
data("Y_proteomic")
X <- X_proteomic[,1:50]
Y <- Y_proteomic
splsicox.model <- splsicox(X, Y, n.comp = 2, penalty = 0.5, x.center = TRUE, x.scale = TRUE)
plot_PLS_Coxmos(splsicox.model, comp = c(1,2), mode = "scores")
```

---

plot_proportionalHazard

plot_proportionalHazard

Description

Generates a visual assessment of the proportional hazards assumption for a given Coxmos model. The function integrates the capabilities of the `survival::cox.zph` and `survminer::ggcoxzph` functions to produce a `ggplot2` graph that visualizes the validity of the proportional hazards assumption.

Usage

```
plot_proportionalHazard(model)
```

Arguments

`model`	Coxmos model.

Details

The proportional hazards assumption is a fundamental tenet of the Cox proportional hazards regression model. It posits that the hazard ratios between groups remain constant over time. Violations of this assumption can lead to biased or misleading results. Thus, assessing the validity of this assumption is crucial in survival analysis.

The function begins by validating the provided model to ensure it belongs to the Coxmos class. If the model is valid, the function then evaluates the proportional hazards assumption using the `survival::cox.zph` function. The results of this evaluation are then visualized using the `survminer::ggcoxzph` function, producing a `ggplot2` graph.

The resulting plot provides a visual representation of the Schoenfeld residuals against time, allowing for an intuitive assessment of the proportional hazards assumption. Each variable or factor level from the model is represented in the plot, and the global test for the proportional hazards assumption is also provided.

This function is instrumental in ensuring the robustness and validity of survival analysis results, offering a comprehensive visualization that aids in the interpretation and validation of the Coxmos model's assumptions.

Value

A `ggplot2` object visualizing the assessment of the proportional hazards assumption for the given Coxmos model. The plot displays the Schoenfeld residuals against time for each variable or factor level from the model. A line is fitted to these residuals to indicate any trend, which can suggest a violation of the proportional hazards assumption.

Author(s)

Pedro Salguero Garcia. Maintainer: pedsalga@upv.edu.es

References

Therneau TM (2024). *A Package for Survival Analysis in R*. R package version 3.5-8, https://CRAN.R-project.org/package=survival. Kassambara A, Kosinski M, Biecek P (2021). *survminer: Drawing Survival Curves using 'ggplot2'*. R package version 0.4.9, https://CRAN.R-project.org/package=survminer. Grambsch PM, Therneau TM (1994). "Proportional hazards tests and diagnostics based on weighted residuals." *Biometrika*. doi:10.1093/biomet/81.3.515, https://academic.oup.com/biomet/article-abstract/81/3/515/257037?redirectedFrom=fulltext. Schoenfeld DA (1982). "Partial residuals for the proportional hazards regression model." *Biometrika*. doi:10.1093/biomet/69.1.239, https://academic.oup.com/biomet/article-abstract/69/1/239/243012?redirectedFrom=fulltext.

Examples

```
data("X_proteomic")
data("Y_proteomic")
X <- X_proteomic[,1:50]
Y <- Y_proteomic
splsicox.model <- splsicox(X, Y, n.comp = 2, penalty = 0.5, x.center = TRUE, x.scale = TRUE)
plot_proportionalHazard(splsicox.model)
```

---

plot_proportionalHazard.list

plot_proportionalHazard.list

Description

Run the function "plot_proportionalHazard" for a list of models. More information in "?plot_proportionalHazard".

Usage

```
plot_proportionalHazard.list(lst_models)
```

Arguments

lst_models	List of Coxmos models.

Value

A ggplot2 object per model visualizing the assessment of the proportional hazards assumption for the given Coxmos model. The plot displays the Schoenfeld residuals against time for each variable or factor level from the model. A line is fitted to these residuals to indicate any trend, which can suggest a violation of the proportional hazards assumption.

Author(s)

Pedro Salguero Garcia. Maintainer: pedsalga@upv.edu.es

Examples

```
data("X_proteomic")
data("Y_proteomic")
X <- X_proteomic[,1:50]
Y <- Y_proteomic
splsicox.model <- splsicox(X, Y, n.comp = 2, penalty = 0.5, x.center = TRUE, x.scale = TRUE)
splsdrcox.model <- splsdrcox(X, Y, n.comp = 2, penalty = 0.5, x.center = TRUE, x.scale = TRUE)
lst_models = list("sPLSICOX" = splsicox.model, "sPLSDRCOX" = splsdrcox.model)
plot_proportionalHazard.list(lst_models)
```

---

plot_pseudobeta *plot_pseudobeta*

Description

This function decomposes a PLS-Cox model, translating it into a pseudo-beta interpretation with respect to the original variables. The decomposition is based on the relationship between the Cox coefficients associated with each component and the weights corresponding to the original variables. The final Cox formula is thus expressed in terms of these original variables.

Usage

```
plot_pseudobeta(
  model,
  error.bar = TRUE,
  onlySig = FALSE,
  alpha = 0.05,
  zero.rm = TRUE,
  top = NULL,
```

```
  auto.limits = TRUE,
  show_percentage = TRUE,
  size_percentage = 3,
  title_size_text = 15,
  legend_size_text = 12,
  x_axis_size_text = 10,
  y_axis_size_text = 10,
  label_x_axis_size = 10,
  label_y_axis_size = 10
)
```

Arguments

model	Coxmos model.
error.bar	Logical. Show error bar (default: TRUE).
onlySig	Logical. Compute pseudobetas using only significant components (default: FALSE).
alpha	Numeric. Numerical values are regarded as significant if they fall below the threshold (default: 0.05).
zero.rm	Logical. Remove variables with a pseudobeta equal to 0 (default: TRUE).
top	Numeric. Show "top" first variables with the higher pseudobetas in absolute value. If top = NULL, all variables are shown (default: NULL).
auto.limits	Logical. If "auto.limits" = TRUE, limits are detected automatically (default: TRUE).
show_percentage	Logical. If show_percentage = TRUE, it shows the contribution percentage for each variable to the full model (default: TRUE).
size_percentage	Numeric. Size of percentage text (default: 3).
title_size_text	Numeric. Text size for legend title (default: 15).
legend_size_text	Numeric. Text size for legend title (default: 12).
x_axis_size_text	Numeric. Text size for x axis (default: 10).
y_axis_size_text	Numeric. Text size for y axis (default: 10).
label_x_axis_size	Numeric. Text size for x label axis (default: 10).
label_y_axis_size	Numeric. Text size for y label axis (default: 10).

Details

The plot_pseudobeta function offers a comprehensive visualization and interpretation of a PLS-Cox model in terms of the original variables. The function begins by validating the model's class and type. For single block models, the function computes the pseudo-betas by multiplying the

loading weights (W.star) with the Cox coefficients. For multiblock models, this computation is performed for each block separately.

The function provides flexibility in terms of visualization. Users can opt to display error bars, filter out non-significant components based on a significance threshold (alpha), and remove variables with a pseudo-beta of zero. Additionally, the function allows for automatic limit detection for the plot and displays the contribution percentage of each variable to the full model. The resulting plot can be customized further with various text size parameters for different plot elements.

It's worth noting that the function supports both single block and multiblock PLS-Cox models. For multiblock models, the function returns a list of plots, one for each block, whereas for single block models, a single plot is returned.

Value

A list containing the following elements: plot: Depending on the model type, this can either be a single ggplot object visualizing the pseudo-beta coefficients for the original variables in a single block PLS-Cox model, or a list of ggplot objects for each block in a multiblock PLS-Cox model. Each plot provides a comprehensive visualization of the pseudo-beta coefficients, potentially including error bars, significance filtering, and variable contribution percentages. beta: A matrix or list of matrices (for multiblock models) containing the computed pseudo-beta coefficients for the original variables. These coefficients represent the influence of each original variable on the survival prediction. sd.min: A matrix or list of matrices (for multiblock models) representing the lower bounds of the error bars for the pseudo-beta coefficients. sd.max: A matrix or list of matrices (for multiblock models) representing the upper bounds of the error bars for the pseudo-beta coefficients.

Author(s)

Pedro Salguero Garcia. Maintainer: pedsalga@upv.edu.es

Examples

```
data("X_proteomic")
data("Y_proteomic")
X <- X_proteomic[,1:50]
Y <- Y_proteomic
splsicox.model <- splsicox(X, Y, n.comp = 2, penalty = 0.5, x.center = TRUE, x.scale = TRUE)
plot_pseudobeta(model = splsicox.model)
```

---

plot_pseudobeta.list *plot_pseudobeta.list*

Description

Run the function "plot_pseudobeta" for a list of models. More information in "?plot_pseudobeta".

Usage

```
plot_pseudobeta.list(
  lst_models,
  error.bar = TRUE,
  onlySig = FALSE,
  alpha = 0.05,
  zero.rm = TRUE,
  top = NULL,
  auto.limits = TRUE,
  show_percentage = TRUE,
  size_percentage = 3,
  verbose = FALSE
)
```

Arguments

lst_models	List of Coxmos models.
error.bar	Logical. Show error bar (default: TRUE).
onlySig	Logical. Compute pseudobetas using only significant components (default: FALSE).
alpha	Numeric. Numerical values are regarded as significant if they fall below the threshold (default: 0.05).
zero.rm	Logical. Remove variables with a pseudobeta equal to 0 (default: TRUE).
top	Numeric. Show "top" first variables with the higher pseudobetas in absolute value. If top = NULL, all variables are shown (default: NULL).
auto.limits	Logical. If "auto.limits" = TRUE, limits are detected automatically (default: TRUE).
show_percentage	Logical. If show_percentage = TRUE, it shows the contribution percentage for each variable to the full model (default: TRUE).
size_percentage	Numeric. Size of percentage text (default: 3).
verbose	Logical. If verbose = TRUE, extra messages could be displayed (default: FALSE).

Value

A list containing the following elements per model: `plot`: Depending on the model type, this can either be a single ggplot object visualizing the pseudo-beta coefficients for the original variables in a single block PLS-Cox model, or a list of ggplot objects for each block in a multiblock PLS-Cox model. Each plot provides a comprehensive visualization of the pseudo-beta coefficients, potentially including error bars, significance filtering, and variable contribution percentages. `beta`: A matrix or list of matrices (for multiblock models) containing the computed pseudo-beta coefficients for the original variables. These coefficients represent the influence of each original variable on the survival prediction. `sd.min`: A matrix or list of matrices (for multiblock models) representing the lower bounds of the error bars for the pseudo-beta coefficients. `sd.max`: A matrix or list of matrices (for multiblock models) representing the upper bounds of the error bars for the pseudo-beta coefficients.

Author(s)

Pedro Salguero Garcia. Maintainer: pedsalga@upv.edu.es

Examples

```
data("X_proteomic")
data("Y_proteomic")
X <- X_proteomic[,1:50]
Y <- Y_proteomic
splsicox.model <- splsicox(X, Y, n.comp = 2, penalty = 0.5, x.center = TRUE, x.scale = TRUE)
splsdrcox.model <- splsdrcox(X, Y, n.comp = 2, penalty = 0.5, x.center = TRUE, x.scale = TRUE)
lst_models = list("sPLSICOX" = splsicox.model, "sPLSDRCOX" = splsdrcox.model)
plot_pseudobeta.list(lst_models = lst_models)
```

---

plot_pseudobeta_newObservation

plot_pseudobeta.newObservation

Description

Generates a visual representation comparing the pseudobeta values derived from the Coxmos model with the values of a new observation. This function provides insights into how the new observation aligns with the established model, offering a graphical comparison of the pseudobeta directions.

Usage

```
plot_pseudobeta_newObservation(
  model,
  new_observation,
  error.bar = TRUE,
  onlySig = TRUE,
  alpha = 0.05,
  zero.rm = TRUE,
  top = NULL,
  auto.limits = TRUE,
  show.betas = FALSE
)
```

Arguments

model	Coxmos model.
new_observation	Numeric matrix or data.frame. New explanatory variables (raw data) for one observation. Qualitative variables must be transform into binary variables.
error.bar	Logical. Show error bar (default: TRUE).
onlySig	Logical. Compute pseudobetas using only significant components (default: TRUE).

alpha	Numeric. Numerical values are regarded as significant if they fall below the threshold (default: 0.05).
zero.rm	Logical. Remove variables with a pseudobeta equal to 0 (default: TRUE).
top	Numeric. Show "top" first variables with the higher pseudobetas in absolute value. If top = NULL, all variables are shown (default: NULL).
auto.limits	Logical. If "auto.limits" = TRUE, limits are detected automatically (default: TRUE).
show.betas	Logical. Show original betas (default: FALSE).

Details

The function plot_pseudobeta.newObservation is designed to visually compare the pseudobeta values from the Coxmos model with those of a new observation. The generated plot is based on the ggplot2 framework and offers a comprehensive view of the relationship between the model's pseudobeta values and the new observation's values.

The function first checks the validity of the provided model and ensures that it belongs to the appropriate class. Depending on the type of the model (either PLS or MB Coxmos methods).

For the actual plotting, the function computes the linear predictor values for the new observation and juxtaposes them with the pseudobeta values from the model. If the show.betas parameter is set to TRUE, the original beta values are also displayed on the plot. Error bars can be included to represent the variability in the pseudobeta values, providing a more comprehensive view of the data's distribution.

The resulting plot serves as a valuable tool for researchers and statisticians to visually assess the alignment of a new observation with an established Coxmos model, facilitating better interpretation and understanding of the data in the context of the model.

Value

A list of four elements: plot: Linear prediction per variable. lp.var: Value of each linear prediction per variable. norm_observation: Observation normalized using the model information. observation: Observation used.

Author(s)

Pedro Salguero Garcia. Maintainer: pedsalga@upv.edu.es

Examples

```
data("X_proteomic")
data("Y_proteomic")
set.seed(123)
index_train <- caret::createDataPartition(Y_proteomic$event, p = .5, list = FALSE, times = 1)
X_train <- X_proteomic[index_train,1:50]
Y_train <- Y_proteomic[index_train,]
X_test <- X_proteomic[-index_train,1:50]
Y_test <- Y_proteomic[-index_train,]
splsicox.model <- splsicox(X_train, Y_train, n.comp = 2, penalty = 0.5, x.center = TRUE,
x.scale = TRUE)
plot_pseudobeta_newObservation(model = splsicox.model, new_observation = X_test[1,,drop=FALSE])
```

plot_pseudobeta_newObservation.list
plot_pseudobeta_newObservation.list

Description

Run the function "plot_pseudobeta_newObservation" for a list of models. More information in "?plot_pseudobeta_newObservation".

Usage

```
plot_pseudobeta_newObservation.list(
  lst_models,
  new_observation,
  error.bar = TRUE,
  onlySig = TRUE,
  alpha = 0.05,
  zero.rm = TRUE,
  top = NULL,
  auto.limits = TRUE,
  show.betas = FALSE,
  verbose = FALSE
)
```

Arguments

lst_models	List of Coxmos models.
new_observation	Numeric matrix or data.frame. New explanatory variables (raw data) for one observation. Qualitative variables must be transform into binary variables.
error.bar	Logical. Show error bar (default: TRUE).
onlySig	Logical. Compute pseudobetas using only significant components (default: TRUE).
alpha	Numeric. Numerical values are regarded as significant if they fall below the threshold (default: 0.05).
zero.rm	Logical. Remove variables with a pseudobeta equal to 0 (default: TRUE).
top	Numeric. Show "top" first variables with the higher pseudobetas in absolute value. If top = NULL, all variables are shown (default: NULL).
auto.limits	Logical. If "auto.limits" = TRUE, limits are detected automatically (default: TRUE).
show.betas	Logical. Show original betas (default: FALSE).
verbose	Logical. If verbose = TRUE, extra messages could be displayed (default: FALSE).

Value

A list of lst_models length with a list of four elements per each model: `plot`: Linear prediction per variable. `lp.var`: Value of each linear prediction per variable. `norm_observation`: Observation normalized using the model information. `observation`: Observation used.

Author(s)

Pedro Salguero Garcia. Maintainer: pedsalga@upv.edu.es

Examples

```
data("X_proteomic")
data("Y_proteomic")
set.seed(123)
index_train <- caret::createDataPartition(Y_proteomic$event, p = .5, list = FALSE, times = 1)
X_train <- X_proteomic[index_train,1:50]
Y_train <- Y_proteomic[index_train,]
X_test <- X_proteomic[-index_train,1:50]
Y_test <- Y_proteomic[-index_train,]
splsicox.model <- splsicox(X_train, Y_train, n.comp = 2, penalty = 0.5, x.center = TRUE,
x.scale = TRUE)
splsdrcox.model <- splsdrcox(X_train, Y_train, n.comp = 2, penalty = 0.5, x.center = TRUE,
x.scale = TRUE)
lst_models = list("sPLSICOX" = splsicox.model, "sPLSDRCOX" = splsdrcox.model)
plot_pseudobeta_newObservation.list(lst_models, new_observation = X_test[1,,drop=FALSE])
```

---

plot_time.list *Time consuming plot.*

Description

Produces a visual representation, using ggplot2, of the computational time consumed by each model encapsulated within the provided list of Coxmos models. This visualization aids in the comparative assessment of computational efficiency across different models.

Usage

```
plot_time.list(lst_models, x.text = "Method", y.text = NULL)
```

Arguments

`lst_models`	List of Coxmos models. Each Coxmos object has the attribute time measured in minutes (cross-validation models could be also added to this function).
`x.text`	Character. X axis title.
`y.text`	Character. Y axis title. If y.text = NULL, then y.text = "Time (mins)" (default: NULL).

Details

The `plot_time.list` function objective is to offer a clear and concise visual representation of the computational time expended by each model during its execution.

The function expects a list of Coxmos models, each of which should inherently possess a time attribute indicating the computational time it consumed. This time attribute is then extracted, aggregated, and visualized in a bar plot format. The function is versatile enough to handle both individual models and cross-validation models, summing up the computational times in the latter case to provide an aggregate view.

The resultant plot is generated using the 'ggplot2' package, ensuring a high-quality and interpretable visualization. The Y-axis of the plot represents the computational time, typically in minutes, while the X-axis enumerates the different models. The function also offers customization options for axis labels, ensuring that the resultant plot aligns with the user's preferences and the intended audience's expectations.

Value

A 'ggplot2' bar plot object.

Author(s)

Pedro Salguero Garcia. Maintainer: pedsalga@upv.edu.es

Examples

```
data("X_proteomic")
data("Y_proteomic")
X <- X_proteomic[,1:50]
Y <- Y_proteomic
coxSW.model <- coxSW(X, Y, x.center = TRUE, x.scale = TRUE)
coxEN.model <- coxEN(X, Y, x.center = TRUE, x.scale = TRUE)
lst_models = list("coxSW" = coxSW.model, "coxEN" = coxEN.model)
plot_time.list(lst_models, x.text = "Method")
```

`predict.Coxmos` *predict.Coxmos*

Description

Generates the prediction score matrix for Partial Least Squares (PLS) Survival models, facilitating the transformation of high-dimensional data into a reduced space while preserving the most relevant information for survival analysis.

Usage

```
## S3 method for class 'Coxmos'
predict(object, ..., newdata = NULL)
```

Arguments

object	Coxmos model
...	additional arguments affecting the predictions produced.
newdata	Numeric matrix or data.frame. New data for explanatory variables (raw data). Qualitative variables must be transform into binary variables.

Details

The predict.Coxmos function is designed to compute the prediction scores for new data based on a previously trained PLS Survival model. The function leverages the dimensional reduction capabilities of PLS to project the new data into a lower-dimensional space, which is particularly beneficial when dealing with high-dimensional datasets in survival analysis. The score matrix obtained serves as a compact representation of the original data, capturing the most salient features that influence survival outcomes.

Value

Score values data.frame for new data using the Coxmos model selected.

Author(s)

Pedro Salguero Garcia. Maintainer: pedsalga@upv.edu.es

Examples

```
data("X_proteomic")
data("Y_proteomic")
set.seed(123)
index_train <- caret::createDataPartition(Y_proteomic$event, p = .5, list = FALSE, times = 1)
X_train <- X_proteomic[index_train,1:50]
Y_train <- Y_proteomic[index_train,]

X_test <- X_proteomic[-index_train,1:50]
Y_test <- Y_proteomic[-index_train,]
model <- splsicox(X_train, Y_train, n.comp = 2) #after CV
predict(object = model, newdata = X_test)
```

---

print.Coxmos *print.Coxmos*

Description

Provides a structured print output for objects of class Coxmos, detailing either the final Cox survival model or the attributes of the optimal model from cross-validation.

Usage

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

Arguments

x	Coxmos object
...	further arguments passed to or from other methods.

Details

The print.Coxmos function serves as a diagnostic tool, offering a comprehensive display of the Coxmos object's attributes. Depending on the nature of the Coxmos object—whether it's derived from a survival model or a cross-validated model—the function tailors its output accordingly. For survival models, it elucidates the method employed, any variables removed due to high correlation, zero or near-zero variance, or non-significance within the Cox model, and presents a summary of the survival model itself. In the context of cross-validated models, the function delineates the cross-validation method utilized and, if ascertainable, details of the best model. For evaluation objects, it systematically enumerates the methods evaluated and provides a summary of metrics for each method.

Value

Print information relative to a Coxmos object.

Author(s)

Pedro Salguero Garcia. Maintainer: pedsalga@upv.edu.es

Examples

```
data("X_proteomic")
data("Y_proteomic")
X <- X_proteomic[,1:50]
Y <- Y_proteomic
model <- splsicox(X, Y, x.center = TRUE, x.scale = TRUE)
print(model)
```

---

save_ggplot *save_ggplot*

Description

Allows to save 'ggplot2' objects in .tiff format based on an specific resolution.

Usage

```
save_ggplot(
  plot,
  folder,
  name = "plot",
  wide = TRUE,
```

```
  quality = "4K",
  dpi = 80,
  custom = NULL
)
```

Arguments

plot	'ggplot2' object. Object to plot and save.
folder	Character. Folder path as character type.
name	Character. File name.
wide	Logical. If TRUE, widescreen format (16:9) is used, in other case (4:3) format.
quality	Character. One of: "HD", "FHD", "2K", "4K", "8K"
dpi	Numeric. Dpi value for the image.
custom	Numeric vector. Custom size of the image. Numeric vector of width and height.

Value

Generate a plot image in the specific folder or working directory.

Author(s)

Pedro Salguero Garcia. Maintainer: pedsalga@upv.edu.es

Examples

```
if(requireNamespace("ggplot2", quietly = TRUE)){
library(ggplot2)
data(iris)
g <- ggplot(iris, aes(Sepal.Width, Sepal.Length, color = Species))
g <- g + geom_point(size = 4)
save_ggplot(g, folder = tempdir())
}
```

---

save_ggplot.svg *save_ggplot.svg*

Description

Allows to save 'ggplot2' objects in .svg format based on an specific resolution.

Usage

```
save_ggplot.svg(
  plot,
  folder,
  name = "plot",
  wide = TRUE,
  quality = "4K",
  dpi = 80,
  custom = NULL
)
```

Arguments

plot	'ggplot2' object. Object to plot and save.
folder	Character. Folder path as character type.
name	Character. File name.
wide	Logical. If TRUE, widescreen format (16:9) is used, in other case (4:3) format.
quality	Character. One of: "HD", "FHD", "2K", "4K", "8K"
dpi	Numeric. Dpi value for the image.
custom	Numeric vector. Custom size of the image. Numeric vector of width and height.

Value

Generate as many plot images as list objects in the specific folder or working directory.

Author(s)

Pedro Salguero Garcia. Maintainer: pedsalga@upv.edu.es

Examples

```
if(requireNamespace("ggplot2", quietly = TRUE)){
library(ggplot2)
data(iris)
g <- ggplot(iris, aes(Sepal.Width, Sepal.Length, color = Species))
g <- g + geom_point(size = 4)
save_ggplot.svg(g, folder = tempdir())
}
```

save_ggplot_lst *save_ggplot_lst*

Description

Allows to save a list of 'ggplot2' objects in .tiff format based on an specific resolution.

Usage

```
save_ggplot_lst(
  lst_plots,
  folder,
  prefix = NULL,
  suffix = NULL,
  wide = TRUE,
  quality = "4K",
  dpi = 80,
  custom = NULL,
  object_name = NULL
)
```

Arguments

lst_plots	List of 'ggplot2'.
folder	Character. Folder path as character type.
prefix	Character. Prefix for file name.
suffix	Character. Sufix for file name.
wide	Logical. If TRUE, widescreen format (16:9) is used, in other case (4:3) format.
quality	Character. One of: "HD", "FHD", "2K", "4K", "8K"
dpi	Numeric. Dpi value for the image.
custom	Numeric vector. Custom size of the image. Numeric vector of width and height.
object_name	Character. If the file to plot it is inside of a list, name of the object to save.

Value

Generate a plot image in the specific folder or working directory.

Author(s)

Pedro Salguero Garcia. Maintainer: pedsalga@upv.edu.es

Examples

```
if(requireNamespace("ggplot2", quietly = TRUE)){
library(ggplot2)
data(iris)
g <- ggplot(iris, aes(Sepal.Width, Sepal.Length, color = Species))
g <- g + geom_point(size = 4)
g2 <- ggplot(iris, aes(Petal.Width, Petal.Length, color = Species))
g2 <- g2 + geom_point(size = 4)
lst_plots <- list("Sepal" = g, "Petal" = g2)
save_ggplot_lst(lst_plots, folder = tempdir())
}
```

---

save_ggplot_lst.svg *save_ggplot_lst.svg*

Description

Allows to save a list of 'ggplot2' objects in .svg format based on an specific resolution.

Usage

```
save_ggplot_lst.svg(
  lst_plots,
  folder,
  prefix = NULL,
  suffix = NULL,
  wide = TRUE,
  quality = "4K",
  dpi = 80,
  custom = NULL,
  object_name = NULL
)
```

Arguments

lst_plots	List of 'ggplot2'.
folder	Character. Folder path as character type.
prefix	Character. Prefix for file name.
suffix	Character. Sufix for file name.
wide	Logical. If TRUE, widescreen format (16:9) is used, in other case (4:3) format.
quality	Character. One of: "HD", "FHD", "2K", "4K", "8K"
dpi	Numeric. Dpi value for the image.
custom	Numeric vector. Custom size of the image. Numeric vector of width and height.
object_name	Character. If the file to plot it is inside of a list, name of the object to save.

Value

Generate as many plot images as list objects in the specific folder or working directory.

Author(s)

Pedro Salguero Garcia. Maintainer: pedsalga@upv.edu.es

Examples

```
if(requireNamespace("ggplot2", quietly = TRUE)){
library(ggplot2)
data(iris)
g <- ggplot(iris, aes(Sepal.Width, Sepal.Length, color = Species))
g <- g + geom_point(size = 4)
g2 <- ggplot(iris, aes(Petal.Width, Petal.Length, color = Species))
g2 <- g2 + geom_point(size = 4)
lst_plots <- list("Sepal" = g, "Petal" = g2)
save_ggplot_lst.svg(lst_plots, folder = tempdir())
}
```

---

sb.splsdrcox *SB.sPLS-DRCOX*

Description

This function performs a single-block sparse partial least squares deviance residual Cox (SB.sPLS-DRCOX). The function returns a Coxmos model with the attribute model as "SB.sPLS-DRCOX".

Usage

```
sb.splsdrcox(
  X,
  Y,
  n.comp = 4,
  penalty = 0.5,
  x.center = TRUE,
  x.scale = FALSE,
  remove_near_zero_variance = TRUE,
  remove_zero_variance = TRUE,
  toKeep.zv = NULL,
  remove_non_significant = FALSE,
  alpha = 0.05,
  MIN_EPV = 5,
  returnData = TRUE,
  verbose = FALSE
)
```

Arguments

X	Numeric matrix or data.frame. Explanatory variables. Qualitative variables must be transform into binary variables.
Y	Numeric matrix or data.frame. Response variables. Object must have two columns named as "time" and "event". For event column, accepted values are: 0/1 or FALSE/TRUE for censored and event observations.
n.comp	Numeric. Number of latent components to compute for the (s)PLS model (default: 10).
penalty	Numeric. Penalty for sPLS-DRCOX. If penalty = 0 no penalty is applied, when penalty = 1 maximum penalty (no variables are selected) based on 'plsRcox' penalty. Equal or greater than 1 cannot be selected (default: 0.5).
x.center	Logical. If x.center = TRUE, X matrix is centered to zero means (default: TRUE).
x.scale	Logical. If x.scale = TRUE, X matrix is scaled to unit variances (default: FALSE).
remove_near_zero_variance	Logical. If remove_near_zero_variance = TRUE, near zero variance variables will be removed (default: TRUE).
remove_zero_variance	Logical. If remove_zero_variance = TRUE, zero variance variables will be removed (default: TRUE).
toKeep.zv	Character vector. Name of variables in X to not be deleted by (near) zero variance filtering (default: NULL).
remove_non_significant	Logical. If remove_non_significant = TRUE, non-significant variables/components in final cox model will be removed until all variables are significant by forward selection (default: FALSE).
alpha	Numeric. Numerical values are regarded as significant if they fall below the threshold (default: 0.05).
MIN_EPV	Numeric. Minimum number of Events Per Variable (EPV) you want reach for the final cox model. Used to restrict the number of variables/components can be computed in final cox models. If the minimum is not meet, the model cannot be computed (default: 5).
returnData	Logical. Return original and normalized X and Y matrices (default: TRUE).
verbose	Logical. If verbose = TRUE, extra messages could be displayed (default: FALSE).

Details

The SB.sPLS-DRCOX function performs a single-block sparse partial least squares deviance residual Cox analysis. This method is designed to handle datasets with a single block of explanatory variables and aims to identify the most relevant features that contribute to the survival outcome. The method combines the strengths of sparse partial least squares (sPLS) with Cox regression, allowing for dimensionality reduction, feature selection, and survival analysis in a unified framework.

The key feature of this function is the use of deviance residuals as the response in the sPLS model. Deviance residuals are derived from a preliminary Cox model and capture the discrepancies between

the observed and expected number of events. By using these residuals as the response, the sPLS model can focus on identifying the explanatory variables that have the most significant impact on the survival outcome.

The function offers flexibility in specifying various hyperparameters, such as the number of latent components (`n.comp`) and the penalty for variable selection (`penalty`). The penalty parameter, `penalty`, controls the sparsity of the model, with higher values leading to more variables being excluded from the model. This allows for a balance between model complexity and interpretability.

Data preprocessing options, such as centering and scaling of the explanatory variables and removal of near-zero variance variables, are also provided. These preprocessing steps ensure that the data is in a suitable format for the sPLS model and can help improve the stability and performance of the analysis.

The output of the function provides a comprehensive overview of the sPLS-DRCOX model, including the normalized data, PLS weights and scores, and the final Cox model. Visualization tools and metrics such as AIC and BIC are also provided to aid in understanding the model's performance and significance of the selected features.

In summary, the `SB.sPLS-DRCOX` function offers a robust approach for survival analysis with high-dimensional data, combining feature selection, dimensionality reduction, and Cox regression in a single-block framework. The method is particularly useful for datasets where the number of variables exceeds the number of observations, and there's a need to identify the most relevant features for predicting survival outcomes.

Value

Instance of class "Coxmos" and model "sb.splscox". The class contains the following elements: `X`: List of normalized X data information.

- `(data)`: normalized X matrix
- `(weightings)`: PLS weights
- `(weightings_norm)`: PLS normalize weights
- `(W.star)`: PLS W* vector
- `(scores)`: PLS scores/variates
- `(x.mean)`: mean values for X matrix
- `(x.sd)`: standard deviation for X matrix

`Y`: List of normalized Y data information.

- `(deviance_residuals)`: deviance residual vector used as Y matrix in the sPLS.
- `(dr.mean)`: mean values for deviance residuals Y matrix
- `(dr.sd)`: standard deviation for deviance residuals Y matrix'
- `(data)`: normalized X matrix
- `(y.mean)`: mean values for Y matrix
- `(y.sd)`: standard deviation for Y matrix'

`survival_model`: List of survival model information.

- `fit`: coxph object.

- AIC: AIC of cox model.
- BIC: BIC of cox model.
- lp: linear predictors for train data.
- coef: Coefficients for cox model.
- YChapeau: Y Chapeau residuals.
- Yresidus: Y residuals.

list_spls_models: List of sPLS-DRCOX models computed for each block.

n.comp: Number of components selected.

penalty Penalty applied.

call: call function

X_input: X input matrix

Y_input: Y input matrix

nzv: Variables removed by remove_near_zero_variance or remove_zero_variance.

nz_coeffvar: Variables removed by coefficient variation near zero.

class: Model class.

time: time consumed for running the cox analysis.

Author(s)

Pedro Salguero Garcia. Maintainer: pedsalga@upv.edu.es

Examples

```
data("X_multiomic")
data("Y_multiomic")
X <- X_multiomic
X$mirna <- X$mirna[,1:50]
X$proteomic <- X$proteomic[,1:50]
Y <- Y_multiomic
sb.splsdrcox(X, Y, n.comp = 2, penalty = 0.5, x.center = TRUE, x.scale = TRUE)
```

---

sb.splsicox *SB.sPLS-ICOX*

Description

This function performs a single-block sparse partial least squares individual Cox (SB.sPLS-ICOX). The function returns a Coxmos model with the attribute model as "SB.sPLS-ICOX".

Usage

```
sb.splsicox(
  X,
  Y,
  n.comp = 4,
  penalty = 1,
  x.center = TRUE,
  x.scale = FALSE,
  remove_near_zero_variance = TRUE,
  remove_zero_variance = TRUE,
  toKeep.zv = NULL,
  remove_non_significant = FALSE,
  alpha = 0.05,
  MIN_EPV = 5,
  returnData = TRUE,
  verbose = FALSE
)
```

Arguments

X	Numeric matrix or data.frame. Explanatory variables. Qualitative variables must be transform into binary variables.
Y	Numeric matrix or data.frame. Response variables. Object must have two columns named as "time" and "event". For event column, accepted values are: 0/1 or FALSE/TRUE for censored and event observations.
n.comp	Numeric. Number of latent components to compute for the (s)PLS model (default: 10).
penalty	Numeric. Penalty for variable selection for the individual cox models. Variables with a lower P-Value than 1 - "penalty" in the individual cox analysis will be keep for the sPLS-ICOX approach (default: 1).
x.center	Logical. If x.center = TRUE, X matrix is centered to zero means (default: TRUE).
x.scale	Logical. If x.scale = TRUE, X matrix is scaled to unit variances (default: FALSE).
remove_near_zero_variance	Logical. If remove_near_zero_variance = TRUE, near zero variance variables will be removed (default: TRUE).
remove_zero_variance	Logical. If remove_zero_variance = TRUE, zero variance variables will be removed (default: TRUE).
toKeep.zv	Character vector. Name of variables in X to not be deleted by (near) zero variance filtering (default: NULL).
remove_non_significant	Logical. If remove_non_significant = TRUE, non-significant variables/components in final cox model will be removed until all variables are significant by forward selection (default: FALSE).

alpha	Numeric. Numerical values are regarded as significant if they fall below the threshold (default: 0.05).
MIN_EPV	Numeric. Minimum number of Events Per Variable (EPV) you want reach for the final cox model. Used to restrict the number of variables/components can be computed in final cox models. If the minimum is not meet, the model cannot be computed (default: 5).
returnData	Logical. Return original and normalized X and Y matrices (default: TRUE).
verbose	Logical. If verbose = TRUE, extra messages could be displayed (default: FALSE).

Details

The SB.sPLS-ICOX function is designed to perform a single-block sparse partial least squares individual Cox analysis. This method is particularly suited for high-dimensional datasets where the number of variables (features) significantly exceeds the number of observations. The "single-block" in its name indicates that while the function can handle datasets with multiple blocks, it processes each block individually rather than in a multiblock manner where all blocks are analyzed simultaneously.

By analyzing one block at a time, the function ensures a focused and detailed examination of each block's contribution to the survival outcome. This approach is especially beneficial when different blocks represent distinct types or sources of data, allowing for a granular understanding of each block's significance.

The analysis begins by applying a penalty to select significant variables based on individual Cox models. This step ensures that only the most relevant features from the current block contribute to the subsequent sPLS analysis. The sPLS method then identifies latent components that capture the maximum covariance between the explanatory variables (X) from the block and the response (Y), which are the deviance residuals from the Cox models.

Users have the flexibility to specify various hyperparameters, including the number of latent components and the penalty for variable selection. The function also offers options for data preprocessing, such as centering, scaling, and removing variables with near-zero or zero variance.

The output provides a comprehensive overview of the analysis for the processed block, including normalized data information, survival model details, and the sPLS-ICOX model. Visualization tools and metrics such as AIC and BIC further aid in understanding the model's performance and significance for the given block.

In summary, the SB.sPLS-ICOX function offers a powerful approach for survival analysis in high-dimensional settings, ensuring optimal feature selection, dimensionality reduction, and predictive modeling for each individual block in the dataset.

Value

Instance of class "Coxmos" and model "sb.splsicox". The class contains the following elements: X: List of normalized X data information.

- (data): normalized X matrix
- (weightings): PLS weights
- (weightings_norm): PLS normalize weights
- (W.star): PLS W* vector

- (scores): PLS scores/variates
- (x.mean): mean values for X matrix
- (x.sd): standard deviation for X matrix

Y: List of normalized Y data information.

- (deviance_residuals): deviance residual vector used as Y matrix in the sPLS.
- (dr.mean): mean values for deviance residuals Y matrix
- (dr.sd): standard deviation for deviance residuals Y matrix'
- (data): normalized X matrix
- (y.mean): mean values for Y matrix
- (y.sd): standard deviation for Y matrix'

survival_model: List of survival model information.

- fit: coxph object.
- AIC: AIC of cox model.
- BIC: BIC of cox model.
- lp: linear predictors for train data.
- coef: Coefficients for cox model.
- YChapeau: Y Chapeau residuals.
- Yresidus: Y residuals.

list_spls_models: List of sPLS-ICOX models computed for each block.

n.comp: Number of components selected.

penalty Penalty applied.

call: call function

X_input: X input matrix

Y_input: Y input matrix

nzv: Variables removed by remove_near_zero_variance or remove_zero_variance.

nz_coeffvar: Variables removed by coefficient variation near zero.

class: Model class.

time: time consumed for running the cox analysis.

Author(s)

Pedro Salguero Garcia. Maintainer: pedsalga@upv.edu.es

Examples

```
data("X_multiomic")
data("Y_multiomic")
X <- X_multiomic
X$mirna <- X$mirna[,1:50]
X$proteomic <- X$proteomic[,1:50]
Y <- Y_multiomic
sb.splsicox(X, Y, n.comp = 2, penalty = 0.5, x.center = TRUE, x.scale = TRUE)
```

splsdacox_dynamic *sPLSDA-COX Dynamic*

Description

The splsdacox_dynamic function conducts a sparse partial least squares discriminant analysis Cox (sPLSDA-COX) using dynamic variable selection methodology. This method is particularly useful for high-dimensional survival data where the goal is to identify a subset of variables that are most predictive of survival outcomes. The function integrates the power of sPLSDA with the Cox proportional hazards model to provide a robust tool for survival analysis in the context of large datasets.

Usage

```
splsdacox_dynamic(
  X,
  Y,
  n.comp = 4,
  vector = NULL,
  MIN_NVAR = 10,
  MAX_NVAR = 1000,
  n.cut_points = 5,
  MIN_AUC_INCREASE = 0.01,
  x.center = TRUE,
  x.scale = FALSE,
  remove_near_zero_variance = TRUE,
  remove_zero_variance = TRUE,
  toKeep.zv = NULL,
  remove_non_significant = FALSE,
  alpha = 0.05,
  EVAL_METHOD = "AUC",
  pred.method = "cenROC",
  max.iter = 200,
  times = NULL,
  max_time_points = 15,
  MIN_EPV = 5,
  returnData = TRUE,
  verbose = FALSE
)
```

Arguments

X	Numeric matrix or data.frame. Explanatory variables. Qualitative variables must be transform into binary variables.
Y	Numeric matrix or data.frame. Response variables. Object must have two columns named as "time" and "event". For event column, accepted values are: 0/1 or FALSE/TRUE for censored and event observations.

`n.comp` Numeric. Number of latent components to compute for the (s)PLS model (default: 10).

`vector` Numeric vector. Used for computing best number of variables. As many values as components have to be provided. If vector = NULL, an automatic detection is perform (default: NULL).

`MIN_NVAR` Numeric. Minimum range size for computing cut points to select the best number of variables to use (default: 10).

`MAX_NVAR` Numeric. Maximum range size for computing cut points to select the best number of variables to use (default: 1000).

`n.cut_points` Numeric. Number of cut points for searching the optimal number of variables. If only two cut points are selected, minimum and maximum size are used. For MB approaches as many as n.cut_points^n.blocks models will be computed as minimum (default: 5).

`MIN_AUC_INCREASE` Numeric. Minimum improvement between different cross validation models to continue evaluating higher values in the multiple tested parameters. If it is not reached for next 'MIN_COMP_TO_CHECK' models and the minimum 'MIN_AUC' value is reached, the evaluation stops (default: 0.01).

`x.center` Logical. If x.center = TRUE, X matrix is centered to zero means (default: TRUE).

`x.scale` Logical. If x.scale = TRUE, X matrix is scaled to unit variances (default: FALSE).

`remove_near_zero_variance` Logical. If remove_near_zero_variance = TRUE, near zero variance variables will be removed (default: TRUE).

`remove_zero_variance` Logical. If remove_zero_variance = TRUE, zero variance variables will be removed (default: TRUE).

`toKeep.zv` Character vector. Name of variables in X to not be deleted by (near) zero variance filtering (default: NULL).

`remove_non_significant` Logical. If remove_non_significant = TRUE, non-significant variables/components in final cox model will be removed until all variables are significant by forward selection (default: FALSE).

`alpha` Numeric. Numerical values are regarded as significant if they fall below the threshold (default: 0.05).

`EVAL_METHOD` Character. If EVAL_METHOD = "AUC", AUC metric will be use to compute the best number of variables. In other case, c-index metric will be used (default: "AUC").

`pred.method` Character. AUC evaluation algorithm method for evaluate the model performance. Must be one of the following: "risksetROC", "survivalROC", "cenROC", "nsROC", "smoothROCtime_C", "smoothROCtime_I" (default: "cenROC").

`max.iter` Numeric. Maximum number of iterations for PLS convergence (default: 200).

times	Numeric vector. Time points where the AUC will be evaluated. If NULL, a maximum of 'max_time_points' points will be selected equally distributed (default: NULL).
max_time_points	Numeric. Maximum number of time points to use for evaluating the model (default: 15).
MIN_EPV	Numeric. Minimum number of Events Per Variable (EPV) you want reach for the final cox model. Used to restrict the number of variables/components can be computed in final cox models. If the minimum is not meet, the model cannot be computed (default: 5).
returnData	Logical. Return original and normalized X and Y matrices (default: TRUE).
verbose	Logical. If verbose = TRUE, extra messages could be displayed (default: FALSE).

Details

The function begins by checking the input parameters for consistency and ensuring that the response variable Y has the required columns "time" and "event". It then preprocesses the data by centering and scaling (if specified), and removing variables with zero or near-zero variance. The function also checks for multicollinearity in the data and addresses it if detected.

The core of the function involves determining the optimal number of variables to retain in the model. If the vector parameter is not provided, the function employs a strategy to identify the best number of variables for each latent component. This is achieved by evaluating different combinations of variables and selecting the one that maximizes the model's performance, as determined by the specified evaluation metric (EVAL_METHOD).

Once the optimal number of variables is determined, the function proceeds to compute the sPLSDA-COX model. It employs the mixOmics::splsda function to compute the sPLSDA model, which is then integrated with the Cox proportional hazards model. The resulting model provides insights into the relationship between the predictor variables and survival outcomes.

The function also offers the flexibility to refine the model further by removing non-significant variables based on a specified alpha threshold.

Value

Instance of class "Coxmos" and model "sPLS-DACOX-Dynamic". The class contains the following elements: `X`: List of normalized X data information.

- `(data)`: normalized X matrix
- `(weightings)`: sPLS weights
- `(W.star)`: sPLS W* vector
- `(loadings)`: sPLS loadings
- `(scores)`: sPLS scores/variates
- `(x.mean)`: mean values for X matrix
- `(x.sd)`: standard deviation for X matrix

`Y`: List of normalized Y data information.

- (data): normalized X matrix
- (y.mean): mean values for Y matrix
- (y.sd): standard deviation for Y matrix'

survival_model: List of survival model information.

- fit: coxph object.
- AIC: AIC of cox model.
- BIC: BIC of cox model.
- lp: linear predictors for train data.
- coef: Coefficients for cox model.
- YChapeau: Y Chapeau residuals.
- Yresidus: Y residuals.

n.comp: Number of components selected.

n.varX: Number of Variables selected in each PLS component.

var_by_component: Variables selected in each PLS component.

plot_accuracyPerVariable: If NULL vector is selected, return a plot for understanding the number of variable selection.

call: call function

X_input: X input matrix

Y_input: Y input matrix

alpha: alpha value selected

nsv: Variables removed by cox alpha cutoff.

nzv: Variables removed by remove_near_zero_variance or remove_zero_variance.

nz_coeffvar: Variables removed by coefficient variation near zero.

class: Model class.

time: time consumed for running the cox analysis.

Author(s)

Pedro Salguero Garcia. Maintainer: pedsalga@upv.edu.es

References

Rohart F, Gautier B, Singh A, Cao KAL (2017). "mixOmics: An R package for 'omics feature selection and multiple data integration." *PLoS Computational Biology*, **13**(11). ISSN 15537358, https://pubmed.ncbi.nlm.nih.gov/29099853/.

Examples

```
data("X_proteomic")
data("Y_proteomic")
X <- X_proteomic[,1:20]
Y <- Y_proteomic
splsdacox_dynamic(X, Y, n.comp = 2, vector = NULL, x.center = TRUE, x.scale = TRUE)
```

splsdrcox *sPLS-DRCOX*

Description

This function performs a sparse partial least squares deviance residual Cox (sPLS-DRCOX) (based on plsRcox R package). The function returns a Coxmos model with the attribute model as "sPLS-DRCOX".

Usage

```
splsdrcox(
  X,
  Y,
  n.comp = 4,
  penalty = 0.5,
  x.center = TRUE,
  x.scale = FALSE,
  remove_near_zero_variance = TRUE,
  remove_zero_variance = FALSE,
  toKeep.zv = NULL,
  remove_non_significant = FALSE,
  alpha = 0.05,
  MIN_EPV = 5,
  returnData = TRUE,
  verbose = FALSE
)
```

Arguments

X	Numeric matrix or data.frame. Explanatory variables. Qualitative variables must be transform into binary variables.
Y	Numeric matrix or data.frame. Response variables. Object must have two columns named as "time" and "event". For event column, accepted values are: 0/1 or FALSE/TRUE for censored and event observations.
n.comp	Numeric. Number of latent components to compute for the (s)PLS model (default: 10).
penalty	Numeric. Penalty for sPLS-DRCOX. If penalty = 0 no penalty is applied, when penalty = 1 maximum penalty (no variables are selected) based on 'plsRcox' penalty. Equal or greater than 1 cannot be selected (default: 0.5).
x.center	Logical. If x.center = TRUE, X matrix is centered to zero means (default: TRUE).
x.scale	Logical. If x.scale = TRUE, X matrix is scaled to unit variances (default: FALSE).
remove_near_zero_variance	Logical. If remove_near_zero_variance = TRUE, near zero variance variables will be removed (default: TRUE).

`remove_zero_variance`
: Logical. If remove_zero_variance = TRUE, zero variance variables will be removed (default: TRUE).

`toKeep.zv`
: Character vector. Name of variables in X to not be deleted by (near) zero variance filtering (default: NULL).

`remove_non_significant`
: Logical. If remove_non_significant = TRUE, non-significant variables/components in final cox model will be removed until all variables are significant by forward selection (default: FALSE).

`alpha`
: Numeric. Numerical values are regarded as significant if they fall below the threshold (default: 0.05).

`MIN_EPV`
: Numeric. Minimum number of Events Per Variable (EPV) you want reach for the final cox model. Used to restrict the number of variables/components can be computed in final cox models. If the minimum is not meet, the model cannot be computed (default: 5).

`returnData`
: Logical. Return original and normalized X and Y matrices (default: TRUE).

`verbose`
: Logical. If verbose = TRUE, extra messages could be displayed (default: FALSE).

Details

The `sPLS-DRCOX` function implements the sparse partial least squares deviance residual Cox (sPLS-DRCOX) model, a specialized approach tailored for survival analysis. This method integrates the strengths of the sparse partial least squares (sPLS) technique with the Cox proportional hazards model, leveraging deviance residuals as a bridge.

The function's core lies in its ability to handle high-dimensional data, often encountered in genomics or other omics studies. By incorporating the `penalty` parameter, which governs the sparsity level, the function offers a fine-grained control over variable selection. This ensures that only the most informative predictors contribute to the model, enhancing interpretability and reducing overfitting.

Data preprocessing is seamlessly integrated, with options to center and scale the predictors, and to remove variables exhibiting near-zero or zero variance. The function also provides a mechanism to retain specific variables, regardless of their variance, ensuring that domain-specific knowledge can be incorporated.

The output is comprehensive, detailing both the sPLS and Cox model components. It provides insights into the selected variables, their contributions across latent components, and the overall fit of the survival model. This rich output aids in understanding the underlying relationships between predictors and survival outcomes.

The `sPLS-DRCOX` function is grounded in established methodologies and is a valuable tool for researchers aiming to unravel complex survival associations in high-dimensional datasets.

Value

Instance of class "Coxmos" and model "sPLS-DRCOX". The class contains the following elements:
`X`: List of normalized X data information.

- `(data)`: normalized X matrix

- (weightings): sPLS weights
- (weightings_norm): sPLS normalize weights
- (W.star): sPLS W* vector
- (loadings): sPLS loadings
- (scores): sPLS scores/variates
- (E): error matrices
- (x.mean): mean values for X matrix
- (x.sd): standard deviation for X matrix

Y: List of normalized Y data information.

- (deviance_residuals): deviance residual vector used as Y matrix in the sPLS.
- (dr.mean): mean values for deviance residuals Y matrix
- (dr.sd): standard deviation for deviance residuals Y matrix'
- (data): normalized X matrix
- (weightings): sPLS weights
- (loadings): sPLS loadings
- (scores): sPLS scores/variates
- (ratio): r value for the sPLS model (used to perform predictions)
- (y.mean): mean values for Y matrix
- (y.sd): standard deviation for Y matrix'

survival_model: List of survival model information.

- fit: coxph object.
- AIC: AIC of cox model.
- BIC: BIC of cox model.
- lp: linear predictors for train data.
- coef: Coefficients for cox model.
- YChapeau: Y Chapeau residuals.
- Yresidus: Y residuals.

penalty: Penalty value selected.

n.comp: Number of components selected.

var_by_component: Variables selected in each PLS component.

call: call function

X_input: X input matrix

Y_input: Y input matrix

B.hat: sPLS beta matrix

R2: sPLS R2

SCR: sPLS SCR

`SCT`: sPLS SCT

`alpha`: alpha value selected

`nsv`: Variables removed by cox alpha cutoff.

`nzv`: Variables removed by remove_near_zero_variance or remove_zero_variance.

`nz_coeffvar`: Variables removed by coefficient variation near zero.

`class`: Model class.

`time`: time consumed for running the cox analysis.

Author(s)

Pedro Salguero Garcia. Maintainer: pedsalga@upv.edu.es

References

Bastien P (2008). "Deviance residuals based PLS regression for censored data in high dimensional setting." *Chemometrics and Intelligent Laboratory Systems*. doi:10.1016/j.chemolab.2007.09.009, https://www.sciencedirect.com/science/article/abs/pii/S0169743907001931?via%3Dihub.
Bastien P, Bastien P, Bertrand F, Meyer N, Meyer N, Meyer N, Maumy-Bertrand M (2015). "Deviance residuals-based sparse PLS and sparse kernel PLS regression for censored data." *Bioinformatics*. https://academic.oup.com/bioinformatics/article/31/3/397/2366078.

Examples

```
data("X_proteomic")
data("Y_proteomic")
X <- X_proteomic[,1:50]
Y <- Y_proteomic
splsdrcox(X, Y, n.comp = 3, penalty = 0.25, x.center = TRUE, x.scale = TRUE)
```

---

splsdrcox_dynamic *sPLS-DRCOX Dynamic*

Description

The sPLS-DRCOX Dynamic function conducts a sparse partial least squares deviance residual Cox regression analysis using a dynamic variable selection approach. This method is particularly useful for high-dimensional survival data where variable selection is crucial. The function returns a model of class "Coxmos" with the attribute model specified as "sPLS-DRCOX".

Usage

```
splsdrcox_dynamic(
  X,
  Y,
  n.comp = 4,
  vector = NULL,
```

```
  MIN_NVAR = 10,
  MAX_NVAR = 1000,
  n.cut_points = 5,
  MIN_AUC_INCREASE = 0.01,
  x.center = TRUE,
  x.scale = FALSE,
  remove_near_zero_variance = TRUE,
  remove_zero_variance = TRUE,
  toKeep.zv = NULL,
  remove_non_significant = FALSE,
  alpha = 0.05,
  EVAL_METHOD = "AUC",
  pred.method = "cenROC",
  max.iter = 200,
  times = NULL,
  max_time_points = 15,
  MIN_EPV = 5,
  returnData = TRUE,
  verbose = FALSE
)
```

Arguments

X	Numeric matrix or data.frame. Explanatory variables. Qualitative variables must be transform into binary variables.
Y	Numeric matrix or data.frame. Response variables. Object must have two columns named as "time" and "event". For event column, accepted values are: 0/1 or FALSE/TRUE for censored and event observations.
n.comp	Numeric. Number of latent components to compute for the (s)PLS model (default: 10).
vector	Numeric vector. Used for computing best number of variables. As many values as components have to be provided. If vector = NULL, an automatic detection is perform (default: NULL).
MIN_NVAR	Numeric. Minimum range size for computing cut points to select the best number of variables to use (default: 10).
MAX_NVAR	Numeric. Maximum range size for computing cut points to select the best number of variables to use (default: 1000).
n.cut_points	Numeric. Number of cut points for searching the optimal number of variables. If only two cut points are selected, minimum and maximum size are used. For MB approaches as many as n.cut_points^n.blocks models will be computed as minimum (default: 5).
MIN_AUC_INCREASE	Numeric. Minimum improvement between different cross validation models to continue evaluating higher values in the multiple tested parameters. If it is not reached for next 'MIN_COMP_TO_CHECK' models and the minimum 'MIN_AUC' value is reached, the evaluation stops (default: 0.01).

`x.center` Logical. If x.center = TRUE, X matrix is centered to zero means (default: TRUE).

`x.scale` Logical. If x.scale = TRUE, X matrix is scaled to unit variances (default: FALSE).

`remove_near_zero_variance`
Logical. If remove_near_zero_variance = TRUE, near zero variance variables will be removed (default: TRUE).

`remove_zero_variance`
Logical. If remove_zero_variance = TRUE, zero variance variables will be removed (default: TRUE).

`toKeep.zv` Character vector. Name of variables in X to not be deleted by (near) zero variance filtering (default: NULL).

`remove_non_significant`
Logical. If remove_non_significant = TRUE, non-significant variables/components in final cox model will be removed until all variables are significant by forward selection (default: FALSE).

`alpha` Numeric. Numerical values are regarded as significant if they fall below the threshold (default: 0.05).

`EVAL_METHOD` Character. If EVAL_METHOD = "AUC", AUC metric will be use to compute the best number of variables. In other case, c-index metric will be used (default: "AUC").

`pred.method` Character. AUC evaluation algorithm method for evaluate the model performance. Must be one of the following: "risksetROC", "survivalROC", "cenROC", "nsROC", "smoothROCtime_C", "smoothROCtime_I" (default: "cenROC").

`max.iter` Numeric. Maximum number of iterations for PLS convergence (default: 200).

`times` Numeric vector. Time points where the AUC will be evaluated. If NULL, a maximum of 'max_time_points' points will be selected equally distributed (default: NULL).

`max_time_points`
Numeric. Maximum number of time points to use for evaluating the model (default: 15).

`MIN_EPV` Numeric. Minimum number of Events Per Variable (EPV) you want reach for the final cox model. Used to restrict the number of variables/components can be computed in final cox models. If the minimum is not meet, the model cannot be computed (default: 5).

`returnData` Logical. Return original and normalized X and Y matrices (default: TRUE).

`verbose` Logical. If verbose = TRUE, extra messages could be displayed (default: FALSE).

Details

The function employs a sparse partial least squares (sPLS) approach combined with deviance residuals from a Cox model to handle survival data. The dynamic variable selection methodology ensures that only the most relevant predictors are included in the model, enhancing interpretability and potentially improving predictive performance.

The input matrices X and Y represent the explanatory and response variables, respectively. It is essential to note that qualitative variables in X should be transformed into binary format. The response matrix Y should have two columns named "time" and "event", where the "event" column can take values 0/1 or FALSE/TRUE, representing censored and event observations.

Several parameters allow users to fine-tune the model. For instance, n.comp determines the number of latent components for the PLS model, and vector aids in computing the optimal number of variables. Parameters like MIN_NVAR and MAX_NVAR define the range for computing cut points to select the best number of variables. The function also provides options for data preprocessing, such as centering and scaling the X matrix and removing variables with near-zero or zero variance.

The evaluation metric for determining the best number of variables can be specified using the EVAL_METHOD parameter. The function supports various evaluation algorithms for assessing model performance, as indicated by the pred.method parameter.

Value

Instance of class "Coxmos" and model "sPLS-DRCOX-Dynamic". The class contains the following elements: `X`: List of normalized X data information.

- `(data)`: normalized X matrix
- `(weightings)`: PLS weights
- `(W.star)`: PLS W* vector
- `(loadings)`: sPLS loadings
- `(scores)`: PLS scores/variates
- `(E)`: error matrices
- `(x.mean)`: mean values for X matrix
- `(x.sd)`: standard deviation for X matrix

`Y`: List of normalized Y data information.

- `(deviance_residuals)`: deviance residual vector used as Y matrix in the sPLS.
- `(dr.mean)`: mean values for deviance residuals Y matrix
- `(dr.sd)`: standard deviation for deviance residuals Y matrix'
- `(data)`: normalized X matrix
- `(y.mean)`: mean values for Y matrix
- `(y.sd)`: standard deviation for Y matrix'

`survival_model`: List of survival model information.

- `fit`: coxph object.
- `AIC`: AIC of cox model.
- `BIC`: BIC of cox model.
- `lp`: linear predictors for train data.
- `coef`: Coefficients for cox model.
- `YChapeau`: Y Chapeau residuals.

- `Yresidus`: Y residuals.

`n.comp`: Number of components selected.

`n.varX`: Number of Variables selected in each PLS component.

`var_by_component`: Variables selected in each PLS component.

`plot_accuracyPerVariable`: If NULL vector is selected, return a plot for understanding the number of variable selection.

`call`: call function

`X_input`: X input matrix

`Y_input`: Y input matrix

`beta_matrix`: PLS beta matrix

`R2`: PLS R2

`SCR`: PLS SCR

`SCT`: PLS SCT

`alpha`: alpha value selected

`nsv`: Variables removed by cox alpha cutoff.

`nzv`: Variables removed by remove_near_zero_variance or remove_zero_variance.

`nz_coeffvar`: Variables removed by coefficient variation near zero.

`class`: Model class.

`time`: time consumed for running the cox analysis.

Author(s)

Pedro Salguero Garcia. Maintainer: pedsalga@upv.edu.es

References

Bastien P (2008). "Deviance residuals based PLS regression for censored data in high dimensional setting." *Chemometrics and Intelligent Laboratory Systems*. doi:10.1016/j.chemolab.2007.09.009, https://www.sciencedirect.com/science/article/abs/pii/S0169743907001931?via%3Dihub.
Bastien P, Bastien P, Bertrand F, Meyer N, Meyer N, Meyer N, Maumy-Bertrand M (2015). "Deviance residuals-based sparse PLS and sparse kernel PLS regression for censored data." *Bioinformatics*. https://academic.oup.com/bioinformatics/article/31/3/397/2366078. Rohart F, Gautier B, Singh A, Cao KAL (2017). "mixOmics: An R package for 'omics feature selection and multiple data integration." *PLoS Computational Biology*, **13**(11). ISSN 15537358, https://pubmed.ncbi.nlm.nih.gov/29099853/.

Examples

```
data("X_proteomic")
data("Y_proteomic")
X <- X_proteomic[,1:50]
Y <- Y_proteomic
splsdrcox_dynamic(X, Y, n.comp = 3, vector = NULL, x.center = TRUE, x.scale = TRUE)
```

splsicox *sPLS-ICOX*

Description

This function performs a sparse partial least squares individual Cox (sPLS-ICOX) (based on plsRcox R package). The function returns a Coxmos model with the attribute model as "sPLS-ICOX".

Usage

```
splsicox(
  X,
  Y,
  n.comp = 4,
  penalty = 0,
  x.center = TRUE,
  x.scale = FALSE,
  remove_near_zero_variance = TRUE,
  remove_zero_variance = FALSE,
  toKeep.zv = NULL,
  remove_non_significant = FALSE,
  alpha = 0.05,
  MIN_EPV = 5,
  returnData = TRUE,
  verbose = FALSE
)
```

Arguments

X	Numeric matrix or data.frame. Explanatory variables. Qualitative variables must be transform into binary variables.
Y	Numeric matrix or data.frame. Response variables. Object must have two columns named as "time" and "event". For event column, accepted values are: 0/1 or FALSE/TRUE for censored and event observations.
n.comp	Numeric. Number of latent components to compute for the (s)PLS model (default: 10).
penalty	Numeric. Penalty for variable selection for the individual cox models. Variables with a lower P-Value than 1 - "penalty" in the individual cox analysis will be keep for the sPLS-ICOX approach (default: 0).
x.center	Logical. If x.center = TRUE, X matrix is centered to zero means (default: TRUE).
x.scale	Logical. If x.scale = TRUE, X matrix is scaled to unit variances (default: FALSE).
remove_near_zero_variance	Logical. If remove_near_zero_variance = TRUE, near zero variance variables will be removed (default: TRUE).

`remove_zero_variance`
: Logical. If remove_zero_variance = TRUE, zero variance variables will be removed (default: TRUE).

`toKeep.zv`
: Character vector. Name of variables in X to not be deleted by (near) zero variance filtering (default: NULL).

`remove_non_significant`
: Logical. If remove_non_significant = TRUE, non-significant variables/components in final cox model will be removed until all variables are significant by forward selection (default: FALSE).

`alpha`
: Numeric. Numerical values are regarded as significant if they fall below the threshold (default: 0.05).

`MIN_EPV`
: Numeric. Minimum number of Events Per Variable (EPV) you want reach for the final cox model. Used to restrict the number of variables/components can be computed in final cox models. If the minimum is not meet, the model cannot be computed (default: 5).

`returnData`
: Logical. Return original and normalized X and Y matrices (default: TRUE).

`verbose`
: Logical. If verbose = TRUE, extra messages could be displayed (default: FALSE).

Details

The `sPLS-ICOX` function is an advanced analytical tool tailored for the elucidation of high-dimensional survival data. It amalgamates the principles of sparse partial least squares (sPLS) regression with individual Cox regression, thereby offering a robust mechanism for both dimension reduction and variable selection in the context of survival analysis. Rooted in the methodologies of the `plsRcox` R package, this function operationalizes the sPLS-ICOX model by leveraging the inherent sparsity introduced via the `penalty` parameter. This parameter delineates a stringent criterion for variable retention, wherein only those variables that manifest a P-Value inferior to the threshold defined by `1 - penalty` in the individual Cox analysis are assimilated into the sPLS-ICOX model framework. The parameter `n.comp` demarcates the number of latent components to be computed for the sPLS model. These latent components, which encapsulate salient patterns within the data, subsequently underpin the Cox regression analysis. It is imperative to underscore the necessity of meticulous data preprocessing, especially in the context of qualitative variables. Such variables necessitate binary transformation prior to their integration into the function. Moreover, the function is equipped with options for data centering and scaling, pivotal operations that can significantly influence model performance. Designed with a predilection for right-censored survival data, the function mandates the structuring of the outcome or response variable `Y` into two distinct columns: "time", which chronicles the survival time, and "event", which catalogues the occurrence or non-occurrence of the event of interest.

Upon execution, the function yields a comprehensive list encapsulating a plethora of elements germane to the sPLS-ICOX model, inclusive of the normalized data matrices, sPLS weight vectors, loadings, scores, and an exhaustive compilation of survival model metrics.

Value

Instance of class "Coxmos" and model "sPLS-ICOX". The class contains the following elements:
`X`: List of normalized X data information.

- `(data)`: normalized X matrix

- (weightings): sPLS weights
- (weightings_norm): sPLS normalize weights
- (W.star): sPLS W* vector
- (loadings): sPLS loadings
- (scores): sPLS scores/variates
- (E): error matrices
- (x.mean): mean values for X matrix
- (x.sd): standard deviation for X matrix

Y: List of normalized Y data information.

- (data): normalized X matrix
- (y.mean): mean values for Y matrix
- (y.sd): standard deviation for Y matrix'

survival_model: List of survival model information.

- fit: coxph object.
- AIC: AIC of cox model.
- BIC: BIC of cox model.
- lp: linear predictors for train data.
- coef: Coefficients for cox model.
- YChapeau: Y Chapeau residuals.
- Yresidus: Y residuals.

n.comp: Number of components selected.

var_by_component: Variables selected by each component.

call: call function

X_input: X input matrix

Y_input: Y input matrix

alpha: alpha value selected

nsv: Variables removed by cox alpha cutoff.

nzv: Variables removed by remove_near_zero_variance or remove_zero_variance.

nz_coeffvar: Variables removed by coefficient variation near zero.

class: Model class.

time: time consumed for running the cox analysis.

Author(s)

Pedro Salguero Garcia. Maintainer: pedsalga@upv.edu.es

References

Bastien P, Vinzi VE, Tenenhaus M (2005). "PLS generalised linear regression." *Computational Statistics & Data Analysis*. https://www.sciencedirect.com/science/article/abs/pii/S0167947304000271?via%3Dihub.

Examples

```
data("X_proteomic")
data("Y_proteomic")
X <- X_proteomic[,1:50]
Y <- Y_proteomic
splsicox(X, Y, n.comp = 2, penalty = 0.5, x.center = TRUE, x.scale = TRUE)
```

`w.starplot.Coxmos` *w.starplot.Coxmos*

Description

The `w.starplot.Coxmos` function offers a graphical representation of the W* (W star) values from a given Coxmos model. Through this visualization, users can gain insights into the variable contributions and their significance in the model. The function provides options for customization, allowing users to focus on specific variables, exclude zero values, and adjust the visual limits.

Usage

```
w.starplot.Coxmos(model, zero.rm = FALSE, top = NULL, auto.limits = TRUE)
```

Arguments

`model`	Coxmos model.
`zero.rm`	Logical. Remove variables equal to 0 (default: FALSE).
`top`	Numeric. Show "top" first variables. If top = NULL, all variables are shown (default: NULL).
`auto.limits`	Logical. If "auto.limits" = TRUE, limits are detected automatically (default: TRUE).

Details

The `w.starplot.Coxmos` function is tailored to visualize the W* values, which are indicative of the variable contributions in a Coxmos model. Initially, the function checks the class of the provided model to ensure its compatibility with the Coxmos framework.

The W* values are extracted from the model and subsequently processed based on user-defined parameters. The `zero.rm` option allows users to exclude variables with zero W* values, ensuring a more concise visualization. If the `top` parameter is specified, the function focuses on displaying only the top-ranked variables based on their absolute W* values.

The visualization is constructed using the 'ggplot2' framework. The color scale can be automatically adjusted to the maximum absolute W* value when the `auto.limits` parameter is set to TRUE. The function also checks for the availability of the `RColorConesa` package. If present, it leverages its color palettes for a more refined visualization; in its absence, default color schemes are applied.

Value

A list of `ggplot2` objects, each representing the W* values for a component of the Coxmos model.

Examples

```
data("X_proteomic")
data("Y_proteomic")
X <- X_proteomic[,1:50]
Y <- Y_proteomic
splsicox.model <- splsicox(X, Y, n.comp = 2, penalty = 0.5, x.center = TRUE, x.scale = TRUE)
w.starplot.Coxmos(model = splsicox.model)
```

X_multiomic *X_multiomic Data*

Description

Toy dataset from BREAST CANCER. miRNA and Protein data. (https://github.com/pilargmarch/multiomics2.0/tree/main)

Usage

```
X_multiomic
```

Format

A data frame with 150 observations and two omics (miRNA and proteomic):

642 miRNAs, 369 proteins

Source

TCGA-BRCA data

X_proteomic *X_proteomic Data*

Description

Toy dataset from BREAST CANCER. Protein data. (https://github.com/pilargmarch/multiomics2.0/tree/main)

Usage

```
X_proteomic
```

Format

A data frame with 150 observations and 369 features:
Small data set from original data (585 observations).

Source

TCGA-BRCA data

Y_multiomic *Y_multiomic Data*

Description

Toy dataset from BREAST CANCER. miRNA and Protein data. (https://github.com/pilargmarch/multiomics2.0/tree/main)

Usage

```
Y_multiomic
```

Format

A data frame with 150 observations and 2 features:

time Global survival time in years. Time to the event of to the last patient information.

event Numeric. FALSE/0 for censored and TRUE/1 for event observations.

Source

TCGA-BRCA data

Y_proteomic *Y_proteomic Data*

Description

Toy dataset from BREAST CANCER. Protein data. (https://github.com/pilargmarch/multiomics2.0/tree/main)

Usage

```
Y_proteomic
```

Format

A data frame with 150 observations and 2 features:

time Global survival time in years. Time to the event of to the last patient information.

event Numeric. FALSE/0 for censored and TRUE/1 for event observations.

Source

TCGA-BRCA data

Index