Package: ePCR (via r-universe)

November 3, 2026

Type Package

Title Ensemble Penalized Cox Regression for Survival Prediction

Version 0.11.0

Date 2024-02-18

Author Teemu Daniel Laajala <teelaa@utu.fi> [aut, cre], Mika Murtojarvi <mianmu2@hotmail.com> [ctb]

Maintainer Teemu Daniel Laajala <teelaa@utu.fi>

Depends R (>= 3.5.0)

Imports grDevices, graphics, stats, methods, glmnet, hamlet, survival, timeROC, pracma, Bolstad2, impute

Suggests MASS, ROCR, c060, utils, Matrix (>= 1.5-0), knitr, rmarkdown

Description The top-performing ensemble-based Penalized Cox Regression (ePCR) framework developed during the DREAM 9.5 mCRPC Prostate Cancer Challenge <https://www.synapse.org/ProstateCancerChallenge> presented in Guinney J, Wang T, Laajala TD, et al. (2017) <doi:10.1016/S1470-2045(16)30560-5> is provided here-in, together with the corresponding follow-up work. While initially aimed at modeling the most advanced stage of prostate cancer, metastatic Castration-Resistant Prostate Cancer (mCRPC), the modeling framework has subsequently been extended to cover also the non-metastatic form of advanced prostate cancer (CRPC). Readily fitted ensemble-based model S4-objects are provided, and a simulated example dataset based on a real-life cohort is provided from the Turku University Hospital, to illustrate the use of the package. Functionality of the ePCR methodology relies on constructing ensembles of strata in patient cohorts and averaging over them, with each ensemble member consisting of a highly optimized penalized/regularized Cox regression model. Various cross-validation and other modeling schema are provided for constructing novel model objects.

License GPL (>= 2)

Encoding UTF-8

RoxygenNote 7.2.3

VignetteBuilder knitr

LazyData true

Config/pak/sysreqs cmake make libicu-dev libuv1-dev

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

Date/Publication 2024-02-19 11:47:55 UTC

RemoteUrl https://github.com/syksy/epcr

RemoteRef HEAD

RemoteSha 629d3b96e004030aeef8f68dd19da61e201eef79

Contents

bootstrapRegCoefs . . . 3
conforminput . . . 4
cv . . . 4
cv.alpha . . . 5
cv.grid . . . 6
DREAM . . . 7
ePCR . . . 8
heatcv . . . 9
integrateRegCurve . . . 10
interact.all . . . 11
interact.part . . . 12
meanrank . . . 12
NelsonAalen . . . 13
normriskrank . . . 14
PEP-class . . . 15
PEP-methods . . . 16
PSP-class . . . 17
PSP-methods . . . 19
score.cindex . . . 21
score.iAUC . . . 22
TimeSurvProb . . . 23
TYKS . . . 24
TYKS_reduced . . . 24
TYKSSIMU . . . 25
zt . . . 26

Index **28**

bootstrapRegCoefs *Bootstrapped testing of regression coefficients in a penalized model*

Description

The purpose of this function is to evaluate a p-value-like statistic for penalized regression coefficients. A fixed number of bootstrapped datasets are generated, and the model coefficients are fitted to these bootstrapped datasets using the pre-determined lambda.

Usage

```
bootstrapRegCoefs(fit, lambda, boot = 1000, epsilon = 10^-6)
```

Arguments

fit	A regularized regression model fit as provided by the glmnet-package
lambda	The pre-fixed corresponding optimal lambda value, typically determined using cross-validation (e.g. cv.glmnet$lambda.1se or cv.glmnet$lambda.min in glmnet)
boot	The number of bootstrapped datasets to generate
epsilon	The tolerance around beta = 0 to still count estimates as zero

Value

Significance values for regression coefficients, defined as the proportion of bootstrapped model fits where coefficient did not shrink within epsilon of zero or where it did not flip sign.

Note

Notice that this is a highly experimental function, and that many statisticians argue that computing p-values does not make sense for penalized models. The null hypothesis is not well defined, as the bias (regularization) pushes the regression coefficients towards zero. Therefore the null hypothesis is not known and the interpretation is not the conventional regression coefficient p-value.

Examples

```
## Not run:
# Computationally too intensive to run bootstrapped fits <5s
data(TYKSSIMU)
library(survival)
x <- as.matrix(xMEDISIMU)
y <- yMEDISIMU[,"surv"]
nlambda <- 30
psp1 <- new("PSP", alphaseq=c(0, 0.5, 1), nlambda = nlambda, folds = 3, x = x, y = y, seeds = 1)
.Object <- psp1
alphaopt <- psp1@optimum["Alpha"]
bs <- bootstrapRegCoefs(fit = psp1@fit, lambda = psp1@optimum["Lambda"], boot = 100)
# Histogram of bootstrapped ps
```

```
hist(bs$ps, breaks=100)

## End(Not run)
```

`conforminput` *Conform the dimensions of a new input data matrix to a readily fitted PEP or PSP object*

Description

Conform the dimensions of a new input data matrix to a readily fitted PEP or PSP object

Usage

```
conforminput(object, newx)
```

Arguments

`object`	A readily fitted PSP or PEP object
`newx`	A data matrix or a data.frame which to expand

Value

An expanded data matrix for which the dimensions conform to the regression coefficients in the PSP or PEP

Author(s)

Teemu Daniel Laajala <teelaa@utu.fi>

`cv` *Function that creates customized cross-validation folds*

Description

The function creates two matrices and returns them as members of a list. 'train' holds training sample indices, columns are cv-folds and rows are sample indices to hold as training samples in each fold. 'test' holds test/validation sample indices, columns are cv-folds and rows are sample indices to hold as test samples in each fold

Usage

```
cv(x, fold = 10, strata = rep(1, times = nrow(x)), shuffle = TRUE, seed = NULL)
```

Arguments

x	The original data matrix or data.frame
fold	Number of desired cross-validation folds; preferably between 3 (3-fold CV) and nrow(x) (LOO-CV)
strata	Indicator if some strata should be balanced over the bins; if no balancing is required, the vector should consist of a single value with length equal to rows in x. Otherwise each strata/batch should be indicated as a unique member in the vector.
shuffle	Whether the indices for the data matrix should be shuffled prior to assigning them to train/test bins
seed	A random seed for reproducibility

Examples

```
data(TYKSSIMU)
cvfolds <- cv(x = xMEDISIMU, fold = 3)
cvfolds$train
cvfolds$test
```

---

cv.alpha *Cross-validation runs for risk predition at a single value of alpha*

Description

Run n-fold cross-validation for a chosen prediction metric at a single value of the L1/L2 norm alpha. A suitable lambda sequence is determined by glmnet, and the cross-validation returns a prediction matrix over the folds over various lambda. This function is mostly called by the higher hierarchy functions, such as cv.grid, which allows varying also the alpha-parameter.

Usage

```
cv.alpha(
  x,
  y,
  folds = 10,
  alpha = 0.5,
  nlamb = 100,
  verb = 0,
  scorefunc,
  plot = FALSE
)
```

Arguments

x	The data matrix to use for predictions
y	The response for coxnet; preferably a preconstructed Surv-object
folds	Number of cross-validation folds
alpha	Chosen L1/L2 norm parameter lambda
nlamb	Number of lambda values
verb	Integer indicating level of verbosity, where 0 is silent and 1 provides additional information
scorefunc	Chosen scoring function, e.g. score.cindex or score.iAUC
plot	Should a CV-performance curve be plotted as a function of lambda, indicating min/max/mean/median of CV performance over the folds

Value

A matrix of cross-validation scores, where rows correspond to CV folds and columns to various lambda values chosen by glmnet

Examples

```
data(TYKSSIMU)
library(survival)
ydat <- Surv(event = yMEDISIMU[,"DEATH"], time = yMEDISIMU[,"LKADT_P"])
set.seed(1)
cvs <- cv.alpha(x = xMEDISIMU, y = ydat, alpha = 0.5, folds = 5,
 nlamb = 50, verb = 1, scorefunc = score.cindex, plot = TRUE)
cvs
```

---

cv.grid *Cross-validation runs for risk predition for a grid of predetermined alpha values and their conditional lambda values*

Description

Expanded Cross-Validation function to run the whole CV in the lambda/alpha grid instead of just lambda-sequence with a pre-specified alpha

Usage

```
cv.grid(
  alphaseq = seq(from = 0, to = 1, by = 0.1),
  seed,
  x,
  y,
  folds = 10,
  nlamb = 100,
```

```
  verb = 0,
  scorefunc,
  plot = FALSE
)
```

Arguments

alphaseq	Sequence of alpha values to test, which should be within [0,1] (with alpha = 0 being ridge regression, 0 < alpha < 1 being elastic net, and alpha = 1 being LASSO)
seed	Random number generation seed for reproducibility
x	Data matrix x
y	The Surv-object response y
folds	Number of folds in the cross-validation
nlamb	Number of lambda values to test in each alpha; notice that these lambda values vary conditional to alpha
verb	Level of verbosity, with 0 as silent and 1 with additional output
scorefunc	Chosen scoring function, e.g. score.cindex or score.iAUC
plot	Whether a performance should be plotted at each varying alpha-value similar to cv.alpha-plots

Value

List of matrices of cross-validation performance values over the alpha/lambda grid for mean/median/min/max/stdev of the chosen performance metric, with rows indicating various alpha-values and columns indicating lambda-values.

Examples

```
data(TYKSSIMU)
library(survival)
ydat <- Surv(event = yMEDISIMU[,"DEATH"], time = yMEDISIMU[,"LKADT_P"])
cvs <- cv.grid(x = xMEDISIMU, y = ydat, folds = 3, nlamb = 30, alphaseq = seq(0, 1, by=5),
 scorefunc = score.iAUC, plot = TRUE, seed = 1)
cvs
```

DREAM *FIMM-UTU DREAM winning implementation of an ensemble of Penalized Cox Regression models for mCPRC research (ePCR)*

Description

FIMM-UTU DREAM winning implementation of an ensemble of Penalized Cox Regression models for mCPRC research (ePCR)

Usage

```
data(ePCRmodels)
```

Format

An object of class PEP of length 1.

Note

Notice that in order to save space, some slots in the S4 object have been set to null.

Author(s)

Teemu Daniel Laajala <teelaa@utu.fi>

References

Guinney J, Wang T, Laajala TD, et al. Prediction of overall survival for patients with metastatic castration-resistant prostate cancer: development of a prognostic model through a crowdsourced challenge with open clinical trial data. Lancet Oncol 2017; 18: 132-142.

---

ePCR *Ensemble Penalized Cox Regression Modeling for Overall Survival and Time-to-Event Prediction in Advanced Prostate Cancer*

Author(s)

Teemu Daniel Laajala <teelaa@utu.fi>

References

Laajala TD, Murtojärvi M, Virkki A, Aittokallio T. ePCR: an R-package for survival and time-to-event prediction in advanced prostate cancer, applied to a real-world patient cohort. Laajala TD, Murtojärvi M, Virkki A, Aittokallio T. ePCR: an R-package for survival and time-to-event prediction in advanced prostate cancer, applied to a real-world patient cohort. Bioinformatics. 2018 Jun 15. doi: 10.1093/bioinformatics/bty477.

Guinney J, Wang T, Laajala TD, et al. Prediction of overall survival for patients with metastatic castration-resistant prostate cancer: development of a prognostic model through a crowdsourced challenge with open clinical trial data. Lancet Oncol 2017; 18: 132-142.

Laajala TD, Guinney J, Costello JC. Community mining of open clinical trial data. Oncotarget 2017; 8: 81721-81722. doi: 10.18632/oncotarget.20853.

`heatcv` *Plot a heatmap of the prediction performance statistic as a function of lambda and alpha combinations*

Description

This function plots a heatmap of cross-validation results by varying the penalization/regularization parameter (lambda, x-axis), together with the corresponding L1/L2 norm parameter alpha (i.e. LASSO, elastic net, ridge regression). The optimal spot in the parameter grid gives insight into the behavior of the regularization in respect to the norms, but note that the lambda-parameter on x-axis is not constant given a conditional alpha-parameter; rather it is a suitable vector chosen by the glmnet-package.

Usage

```
heatcv(
  psp,
  bias = 0.1,
  by.rownames = 1,
  by.colnames = 1,
  paletcol = c("cyan", "blue", "black", "red", "orange"),
  paletncol = 1000,
  xlab = "Alpha-dependent log-Lambda",
  ylab = "Alpha",
  main = "",
  plot.opt = TRUE,
  plot.1sd = FALSE,
  ...
)
```

Arguments

`psp`	An S4-class PSP-object to plot, as built using the ePCR-package
`bias`	Bias in color palette (skews it to favor distinguishing high values better by default)
`by.rownames`	Show every n:th row name (helps for dense axis labels)
`by.colnames`	Show every n:th column name (helps for dense axis labels)
`paletcol`	Names for colours to include in the heatmap palette
`paletncol`	Number of colours on the color key
`xlab`	Label for the x-axis (typically log-lambda penalization parameter)
`ylab`	Label for the y-axis (typically alpha-value indicating LASSO, elastic net or ridge regression)
`main`	Main label on top of the heatmap
`plot.opt`	Should the best (highest) performance statistic be indicated as a large dot on the heatmap

plot.1sd	Should boundaries of the optimal performance statistic area be outlined as within 1 standard deviation of the optimal spot (note: experimental). This attempts to mimic the 1sd-optimum suggested in the glmnet-package for cross-validation for a constant alpha parameter but for 2 dimensions.
...	additional parameters passed on to the hmap-function of hamlet-package

Note

The heatmap plotting is compatible with the default plot-region in a R graphic canvas. The function hmap from the same author's hmap-package can be highly customized to fit more specific needs.

Author(s)

Teemu Daniel Laajala <teelaa@utu.fi>

Examples

```
data(ePCRmodels)
par(mfrow=c(1,3))
heatcv(DREAM@PSPs[[1]], main=DREAM@PSPs[[1]]@description, by.rownames=10, by.colnames=10)
heatcv(DREAM@PSPs[[2]], main=DREAM@PSPs[[2]]@description, by.rownames=10, by.colnames=10)
heatcv(DREAM@PSPs[[3]], main=DREAM@PSPs[[3]]@description, by.rownames=10, by.colnames=10)
```

---

integrateRegCurve *Integrate the area over/under the regularization path of a penalized regression model*

Description

This function evaluates the overall significance of a regularized regression coefficient in a penalized Cox model. It takes into account the whole range of lambda-penalization parameter, and computes the area over or under the regularization curve. This gives more insight into the importance of a regression coefficient over the whole range of lambda, instead of evaluating it at a single optimal lambda point determined typically using cross-validation.

Usage

```
integrateRegCurve(fit, weighted = FALSE)
```

Arguments

fit	A regularized regression model fited using glmnet
weighted	Should the regularization curve be weighted by the corresponding lambda (as higher lambda pushes coefficients to zero)

Value

Integrated area over or under a regularization curve using the trapezoid method from the pracma-package

Examples

```
# Exemplify one PSP of the readily fitted ensembles
data(ePCRmodels)
RegAUC <- cbind(
integrateRegCurve(fit = DREAM@PSPs[[1]]@fit),
integrateRegCurve(fit = DREAM@PSPs[[2]]@fit),
integrateRegCurve(fit = DREAM@PSPs[[3]]@fit)
)
SortRegAUC <- RegAUC[order(apply(RegAUC, MARGIN=1,
 FUN=function(z) abs(mean(z)) ), decreasing=TRUE),]
colnames(SortRegAUC) <- c(DREAM@PSPs[[1]]@description,
DREAM@PSPs[[2]]@description,
DREAM@PSPs[[3]]@description)
SortRegAUC[1:10,] # Top 10 coefficients according to (absolute) regularization curve auc
```

`interact.all` *Compute all pairwise interactions between the columns of a data matrix*

Description

The function multiplies the columns (variables) of a matrix or a data.frame with each other, and produces a new matrix where all pairwise interactions are present. This also includes multiplying a column with its self, thus effectively returning a squared column.

Usage

```
interact.all(input)
```

Arguments

`input`	A data matrix (of class matrix or data.frame) for which all column-wise multiplications are to be computed

Value

A matrix where columns of the original data matrix have been multiplied, indicating column names coupled with a colon in-between

Examples

```
set.seed(1)
somedata <- data.frame(a = rnorm(10), b = rnorm(10), c = runif(10), d = runif(10))
somedata
allinteract <- interact.all(somedata)
allinteract
```

interact.part — *Compute a chosen set of pairwise interactions between two sets of columns in a data matrix*

Description

Similar to interact.all-function, but here user provides two sets of variables, and each pairwise combination between these two sets is multiplied. These pairwise interactions are then returned as a new data matrix, with a colon indicating which variables were multiplied.

Usage

```
interact.part(input, first, second)
```

Arguments

input	The input data matrix, of either class matrix or data.frame
first	The first set of columns to combine with each of the members of the second set, as either integers or column names
second	The second set of columns to combine with each of the members of the first set, as either integers or column names

Value

A data matrix with multiplied columns as indicated using the sets 'first' and 'second'

Examples

```
set.seed(1)
somedata <- data.frame(a = rnorm(10), b = rnorm(10), c = runif(10), d = runif(10))
somedata
someinteract <- interact.part(somedata, first = c("a", "b"), second = c("c", "d"))
someinteract
```

meanrank — *Compute mean of predicted risk ranks for an ePCR ensemble*

Description

Compute mean of predicted risk ranks for an ePCR ensemble

Usage

```
meanrank(x)
```

Arguments

x	A list or a matrix of risk scores given per each ensemble member (each column or list member is considered an equal member of the ensemble)

Value

An averaged predicted risk rank over all the ensemble members

Note

Extensively called by the 'predict'-function for PEP-objects when risk predictions are performed over the ensemble

Author(s)

Teemu Daniel Laajala <teelaa@utu.fi>

NelsonAalen *Cox-Oakes extension of the Nelson-Aalen estimates for a Cox model*

Description

Implementing the heuristic Cox and Oakes extension of the Nelson-Aalen estimate for Cox model to extract individual-specific survival. Time-to-event predictions are then given at the first time point at which an individual reaches an event probability of 50

Usage

```
NelsonAalen(
  b,
  Xold,
  Xnew,
  events,
  time,
  tpred = 0:round(max(time, na.rm = T), 0),
  plot = FALSE
)
```

Arguments

b	Beta coefficients at optimal glmnet coxnet model (lambda, alpha)
Xold	Data matrix with rows as individuals (i.e. training data)
Xnew	Possible new prediction data matrix (if omitted the training data is used, or if it is a new dataset the columns should comform to the training data and new individuals be provided as rows)
events	Deaths or right-censoring per each individual (1 death 0 alive censored) for Xold

time	Times to event or censoring for Xold
tpred	The predicted time points; more tight grid gives a smoother curve
plot	Should an individualized plot be plotted to show how the cumulative survival curves behave

Value

Predicted times-to-event predictions either for the training data or an optional provided novel dataset

Note

See section 3.6 at http://data.princeton.edu/pop509/NonParametricSurvival.pdf for the reference

Author(s)

Teemu Daniel Laajala <teelaa@utu.fi>

Examples

```
data(TYKSSIMU)
library(survival)
xdat <- as.matrix(xMEDISIMU)
ydat <- yMEDISIMU[,"surv"]
```

---

normriskrank *Normalize ensemble risk scores to ranks and then to uniform range*

Description

Normalize ensemble risk scores to ranks and then to uniform range

Usage

```
normriskrank(x)
```

Arguments

x	A list or a matrix of risk scores given per each ensemble member (each column or list member is considered an equal member of the ensemble

Value

An averaged predicted risk rank over all the ensemble members that has been normalized to the range [0,1] based on: (x - min(x)) / (max(x) - min(x)) -> [0,1]

Note

Normalizes 'predict'-function calls for PEP-objects after calling 'meanrank'-function

Author(s)

Teemu Daniel Laajala <teelaa@utu.fi>

PEP-class *Penalized Ensemble Predictor (PEP) S4-class ensemble consisting of individual PSP-members*

Description

This class constructs an ensemble of individual Penalized Ensemble Predictor (PSP-class) members. Each member contributes to the model output equally, and ensemble-level functions wrap up individual predictions into an averaged ensemble prediction. The user may define an arbitrary number of PSPs and tailor them to suit the particular needs, and then provide them as a list to the PEP-constructor. As such, constructing well tailored individual ensemble members (of PSP-class) in order to produce a powerful ensemble (of PEP-class) is important on both levels.

Slots

`PSPs` List of PSP-objects that will be treated as equal members of the ensemble

`description` A character string describing the structure or purpose of the ensemble

`features` A character list of variable/feature names

`dictionary` A named list of above variables/features and their more precise description

`predens` A function for compiling all predictions from the PSPs into consensus prediction

`prednorm` A function for normalizing the predictions e.g. to risk scores in [0,1]

Examples

```
## Not run:
# The PEP-construction is wrapped in NOT RUN, because cross-validating multiple PSPs
# is very time consuming especially if a tight grid of alpha/lambda is to be explored.
# The simulated data from Turku University Hospital (TYKS) is used as an example:
data(TYKSSIMU)

# Two cohorts and corresponding data matrices:
head(xMEDISIMU)
head(xTEXTSIMU)
# Two survival responses:
head(yMEDISIMU)
head(xTEXTSIMU)

# Search L1/L2 norm alpha-grid with 10 values between [0,1]
aseq <- seq(from=0, to=1, by=0.1)
# Lambda sequence penalization is of 100 length conditional for each alpha
nlamb <- 100

library(survival)
# Create three ensemble members; one for MEDI cohort, one for TEXT cohort,
```

```
# and finally one member that combines both cohorts simultaneously in a coxnet
psp1 <- new("PSP", x = rbind(xMEDISIMU, xTEXTSIMU),
y = Surv(rbind(yMEDISIMU, yTEXTSIMU)[,"surv"]),
plot = TRUE, alphaseq = aseq, scorefunc = score.cindex, seed = 1,
folds = 10, nlambda = nlamb)
psp2 <- new("PSP", x = xMEDISIMU,
y = Surv(yMEDISIMU[,"surv"]),
plot = TRUE, alphaseq = aseq, scorefunc = score.cindex, seed = 1,
folds = 10, nlambda = nlamb)
psp3 <- new("PSP", x = xTEXTSIMU,
 y = Surv(yTEXTSIMU[,"surv"]),
plot = TRUE, alphaseq = aseq, scorefunc = score.cindex, seed = 1,
folds = 10, nlambda = nlamb)
par(mfrow=c(1,3))
plot(psp1); plot(psp2); plot(psp3); # Inspect the alpha/lambda surfaces

# Create an ensemble of the above 3 members
simuens <- new("PEP", PSPs = list(psp1, psp2, psp3))
simuens
# Ready PEP-object can be used for novel predictions etc


## End(Not run)

# Run example predictions from a previously optimized PEP-model
data(ePCRmodels)
data(TYKSSIMU)

# Perform risk predictions from the joint cohort ensemble member as an example
MEDIpred <- predict(TYKS@PSPs[[1]]@fit, s=TYKS@PSPs[[1]]@optimum["Lambda"],
newx = conforminput(TYKS@PSPs[[1]], xMEDISIMU))[,1]
TEXTpred <- predict(TYKS@PSPs[[1]]@fit, s=TYKS@PSPs[[1]]@optimum["Lambda"],
newx = conforminput(TYKS@PSPs[[1]], xTEXTSIMU))[,1]

# Risk scores obtained for the new patients (arbitrary unit as per Cox regression)
head(MEDIpred)
head(TEXTpred)
```

PEP-methods *PEP-methods*

Description

PEP-methods

print.PEP: Print a PEP object to the console

predict.PEP: Predict for a novel patient from current PEP-ensemble

Usage

```
## S4 method for signature 'PEP'
print(x, ...)

## S4 method for signature 'PEP'
predict(object, type = "response", newx, x.expand)
```

Arguments

x	Generic x
...	Additional custom parameters passed on
object	PEP-ensemble model object
type	Type of prediction; either "response" or "ensemble"
newx	New data matrix
x.expand	A function that may expand (i.e. extract features) from the input data matrix. By default this will be the default x.expand saved in the S4-slot of the first ensemble member. If the user wishes to omit this functionality, setting this parameter to 'x.expand = as.matrix' does not expand the input data matrix. Notice that if the user has manually called the 'conforminput' function for the newx-data, it is no longer necessary to expand the data matrix here.

PSP-class *Penalized Single Predictor (PSP) S4-class as a member of PEP-ensembles*

Description

PSP is a single penalized Cox regression model, where an alpha/lambda grid has been optimized using cross-validation and a chosen prediction metric. PSPs are single entities that will compile together into PEPs, the ensemble objects that will average over multiple PSPs to generate an ensemble prediction. Typically a single PSP models a part of the data, such as a cohort strata.

Slots

`description` A general user-provided string describing the PSP

`features` A character vector indicating feature names

`strata` Information whether data matrix x included substrata (will be used in plotting functions etc)

`alphaseq` The sequence of alpha values to test, ranging between [0,1]; alpha = 0 being ridge regression, 0 < alpha < 1 being elastic net and alpha = 1 being LASSO

`cvfolds` The number of cross-validation folds to utilize; by default 10

`nlambda` The amount of lambda values utilized in each regularization path; by default 100 as in glmnet-package

`cvmean` A matrix indicating the mean CV performance in alpha/lambda grid (preferred over median)

`cvmedian` A matrix indicating the median CV performance in alpha/lambda grid

`cvstdev` A matrix indicating the standard deviation in CV performance over the folds in the alpha/lambda grid

`cvmin` A matrix indicating minimum CV performance in alpha/lambda grid

`cvmax` A matrix indicating maximum CV performance in alpha/lambda grid

`score` The scoring function, user-defined or one provided by ePCR package such as score.cindex or score.iAUC

`cvrepeat` Number of cross-validation procedures to run multiple times and then average over, in order to reduce the effect of binning samples

`impute` The imputation function used if provided matrix 'x' includes missing values; by default the impute.knn-function from BioConductor package 'impute'

`optimum` The optimum in alpha/lambda grid, with optimal alpha and similarly for lambda

`seed` The initial random seed used for cross-validation

`x` The input data matrix

`x.expand` A function that allows expansion of matrix 'x' to include interactions between variables; if no such are desired, this should be an identity function

`y` The Surv-object as in survival-package, which serves as the response y

`fit` The glmnet coxnet-object obtained with optimal alpha

`criterion` The optimizing criterion; by default "min" for minimizing CV-error

`dictionary` A list of discriptions for each variable

`regAUC` A numeric vector for the AUC under regularization curve as computed by integrateRegCurve-function

Examples

```
# As an example, illustrate a naive PSP built on the small medication cohort
data(TYKSSIMU)
library(survival)
# Minimal example with much fewer patients and variables
psp_ex <- new("PSP", alphaseq=c(0.2, 0.8), nlambda=20, folds=3,
 x = xMEDISIMU[1:80,c(1:20,40:50)], y = yMEDISIMU[1:80,"surv"],
seeds = 1, score=score.cindex)

plot(psp_ex) # Optimization surface of alpha/lambda

# Illustrate the use of some PSP-methods:
PSP.KM(psp_ex, cutoff = 0.5) # Kaplan-Meier
PSP.PCA(psp_ex) # PCA plot of training data
PSP.BOX(psp_ex) # Boxplots, here for the first training variable
PSP.CSP(psp_ex) # Cumulative survival probabilities for the training data
invisible(PSP.NA(psp_ex)) # Time-to-event Nelson-Aalen heuristic algorithm

## Not run:
```

```
# Computationally intensive novel PSP-fitting is omitted from the test runs
# Functions for readily fitted PSP-objects are illustrated above
data(TYKSSIMU)
library(survival)
psp_meditext <- new("PSP", x = rbind(xMEDISIMU, xTEXTSIMU),
y = Surv(rbind(yMEDISIMU, yTEXTSIMU)[,"surv"]),
plot = TRUE, alphaseq = seq(0, 1, by=.01), scorefunc = score.cindex,
seed = 1, folds = 10, nlambda = 100)
plot(psp_meditext)

## End(Not run)
```

PSP-methods *PSP-methods*

Description

PSP-methods

print.PSP: Print general information of PSPs contents to the terminal

plot.PSP: By default the mean CV surface in terms of alpha/lambda is plotted using hamlet-package's hmap-function

coef.PSP: Default PSP coef-function extracts only the optimum parameters, not whole lambda-range

predict.PSP: Predict for a novel patient from current PSP

PSP.KM: Kaplan-Meier with division at a given cutoff point within [0,1]

PSP.PCA: Principal Component Plot of a single PSP, showing 2 principal axes with a colouring if strata have been indicated; newx can also be plotted in relation to fitting data

PSP.BOX: Boxplot of a single variable in a PSP in respect to strata, for outlier detection and observing variable distributions

PSP.CSP: Cumulative survival probabilities

PSP.NA: Nelson-Aalen with time-to-event prediction at point t = F^-1(0.5)

Usage

```
## S4 method for signature 'PSP'
print(x, ...)

plot(x, y, ...)

## S4 method for signature 'PSP'
plot(x, y, bias = 0.1, ...)

## S4 method for signature 'PSP'
coef(object)
```

```
## S4 method for signature 'PSP'
predict(object, type = "response", newx, verb = 0)

PSP.KM(object, ...)

## S4 method for signature 'PSP'
PSP.KM(object, cutoff = 0.5)

PSP.PCA(object, ...)

## S4 method for signature 'PSP'
PSP.PCA(
  object,
  newx,
  expanded = TRUE,
  type = "all",
  shuffle = TRUE,
  z = TRUE,
  cex = 1,
  col = c("aquamarine", "coral", "royalblue", "black"),
  pch = 16
)

PSP.BOX(object, ...)

## S4 method for signature 'PSP'
PSP.BOX(object, newx, var = colnames(object@x)[1], expanded = FALSE)

PSP.CSP(object, ...)

## S4 method for signature 'PSP'
PSP.CSP(object, newx, t = seq(from = 1, to = 36 * 30.5, by = 1), plot = FALSE)

PSP.NA(object, ...)

## S4 method for signature 'PSP'
PSP.NA(object, newx, plot = TRUE)
```

Arguments

x	Generic x
...	Additional custom parameters passed on
y	Generic y
bias	Bias for skewing the color in heatmap key plotting
object	PSP-object
type	Types of variables to include; recognizes (int)eger, (bin)ary and (num)eric
newx	New data matrix

verb	Level of verbosity
cutoff	Cutoff point for division
expanded	Should data matrix expansion through interactions be included
shuffle	Shuffle plotting order
z	Should data centering and scaling should be conducted
cex	Zooming multiplier
col	Vector of color numbers or names to use for strata
pch	Point type to use (refer to par-function pch-parameter)
var	Name of variable to plot
t	Sequence of time points to evaluate cumulative survival probabilities at
plot	Plot the corresponding functionality

Note

Please refer to the PSP-class examples for applying these PSP-methods

---

score.cindex *Scoring function for evaluating survival prediction through concordance index (c-index)*

Description

C-index (Concordance index) of the predicted vs. true answer, i.e. proportion of pairs that go in correct direction over all pairwise comparisons

Usage

```
score.cindex(pred, time, event, real)
```

Arguments

pred	Numeric risk score for each event
time	A vector of event or censoring times
event	A binary valued vector that indicates either death (1) or right-censoring (0)
real	A previously constructed Surv-object instead of providing time and event

Value

Corcordance index (c-index) of the prediction

Examples

```
# A random prediction ought to be near 0.5
# c-index is not sensitive to time scale, as it tests pairwise prediction accuracy
set.seed(1); prediction <- sample(1:20)
time <- seq(from=1000, to=50, by=-50)
event <- c(0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1)
library(survival)
score.cindex(pred = prediction, real = Surv(time=time, event=event))
```

`score.iAUC` *Scoring function for evaluating survival prediction by time-wise integrated AUC*

Description

Time-wise integrated prediction for survival is performed by this scoring function using the timeROC-package. It's offered as an alternative to the score.cindex-function with the difference that time-wise integrated AUC is sensitive to the choice of time-window. By default (as similar to DREAM 9.5 mCRPC challenge), the AUCs are determined at 6 to 30 months, and the AUC is then normalized to a score within [0,1]. Notice that for studies shorter or longer than this proposed time window, the scoring function should be adjusted accordingly.

Usage

```
score.iAUC(pred, time, event, real, times = seq(6, 30, by = 1) * 30.5)
```

Arguments

`pred`	Numeric risk score for each event
`time`	A vector of event or censoring times
`event`	A binary valued vector that indicates either death (1) or right-censoring (0)
`real`	A previously constructed Surv-object instead of providing time and event
`times`	Time-points at which to evaluate the iAUC

Value

The integrated area under the ROC-curve over time

Examples

```
# A random prediction ought to be near 0.5
# iAUC is sensitive to the choice of time points to test AUC at
set.seed(1); prediction <- sample(1:20)
time <- seq(from=1000, to=50, by=-50)
event <- c(0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1)
library(survival)
score.iAUC(pred = prediction, real = Surv(time=time, event=event))
```

`TimeSurvProb` *Predict cumulative survival probabilities for new data at given time points*

Description

Given a readily fitted regularized Cox regression model, this function predicts the cumulative survival probabilities for new data at time points determined by the user. The function uses c060-package's functionality for computing base hazard, and then performs linear predictions for new observations using the fitted regularized Cox regression model.

Usage

```
TimeSurvProb(
  fit,
  time,
  event,
  olddata,
  newdata,
  s,
  times = c(1:36) * 30.5,
  plot = FALSE
)
```

Arguments

`fit`	A single regularized Cox regression model fitted using glmnet
`time`	Time to events for the training data
`event`	Event indicators for the training data (0 censored, 1 event)
`olddata`	The old data matrix used to fit the original 'fit' glmnet-object
`newdata`	The new data matrix for which to predict time-to-event prediction (should comform to the old data matrix)
`s`	The optimal lambda parameter as used in the glmnet-package for its fit objects
`times`	The time points at which to estimate the cumulative survival probabilities (by default in days)
`plot`	Should the cumulative survival probabilities be plotted as a function of time

Value

Cumulative survival probabilities at the chosen time points

Author(s)

Teemu Daniel Laajala <teelaa@utu.fi>

TYKS *ePCR model fitted to the Turku University Hospital cohorts (all features)*

Description

ePCR model fitted to the Turku University Hospital cohorts (all features)

Usage

```
data(ePCRmodels)
```

Format

An object of class PEP of length 1.

Note

Notice that in order to save space, some slots in the S4 object have been set to null.

Author(s)

Teemu Daniel Laajala <teelaa@utu.fi>

References

Laajala TD, Murtojärvi M, Virkki A, Aittokallio T. ePCR: an R-package for survival and time-to-event prediction in advanced prostate cancer, applied to a real-world patient cohort. Laajala TD, Murtojärvi M, Virkki A, Aittokallio T. ePCR: an R-package for survival and time-to-event prediction in advanced prostate cancer, applied to a real-world patient cohort. Bioinformatics. 2018 Jun 15. doi: 10.1093/bioinformatics/bty477.

TYKS_reduced *ePCR model fitted to the Turku University Hospital cohorts (features derived from text mining only)*

Description

ePCR model fitted to the Turku University Hospital cohorts (features derived from text mining only)

Usage

```
data(ePCRmodels)
```

Format

An object of class PEP of length 1.

Note

Notice that in order to save space, some slots in the S4 object have been set to null.

Author(s)

Teemu Daniel Laajala <teelaa@utu.fi>

References

Laajala TD, Murtojärvi M, Virkki A, Aittokallio T. ePCR: an R-package for survival and time-to-event prediction in advanced prostate cancer, applied to a real-world patient cohort. Laajala TD, Murtojärvi M, Virkki A, Aittokallio T. ePCR: an R-package for survival and time-to-event prediction in advanced prostate cancer, applied to a real-world patient cohort. Bioinformatics. 2018 Jun 15. doi: 10.1093/bioinformatics/bty477.

TYKSSIMU *TYKSSIMU - simulated data matrices and survival responses from Turku University Hospital*

Description

TYKSSIMU - simulated data matrices and survival responses from Turku University Hospital

xMEDISIMU: Simulated prostate cancer data from Turku University Hospital (data matrix x, Medication-cohort)

xTEXTSIMU: Simulated prostate cancer data from Turku University Hospital (data matrix x, Text-cohort)

yMEDISIMU: Simulated prostate cancer data from Turku University Hospital (survival response y, Medication-cohort)

yTEXTSIMU: Simulated prostate cancer data from Turku University Hospital (survival response y, Text-cohort)

Usage

```
data(TYKSSIMU)

xMEDISIMU

xTEXTSIMU

yMEDISIMU

yTEXTSIMU
```

Format

xTEXTSIMU:

xMEDISIMU:

yTEXTSIMU:

yMEDISIMU:

An object of class `data.frame` with 150 rows and 101 columns.

An object of class `data.frame` with 500 rows and 101 columns.

An object of class `data.frame` with 150 rows and 3 columns.

An object of class `data.frame` with 500 rows and 3 columns.

Author(s)

Teemu Daniel Laajala <teelaa@utu.fi>, Mika Murtojärvi <mianmu2@hotmail.com>

References

Laajala TD, Murtojärvi M, Virkki A, Aittokallio T. ePCR: an R-package for survival and time-to-event prediction in advanced prostate cancer, applied to a real-world patient cohort. Laajala TD, Murtojärvi M, Virkki A, Aittokallio T. ePCR: an R-package for survival and time-to-event prediction in advanced prostate cancer, applied to a real-world patient cohort. Bioinformatics. 2018 Jun 15. doi: 10.1093/bioinformatics/bty477.

Examples

```
data(TYKSSIMU)
head(xTEXTSIMU)
head(xMEDISIMU)
head(yTEXTSIMU)
head(yMEDISIMU)
dim(xTEXTSIMU)
dim(xMEDISIMU)
```

---

zt

Extended function for z-transformation, filling non-finite values and changes column names at will

---

Description

An extended function of the standard z-score standardization of a vector in R (i.e. function 'scale'). Supports filling in non-finite values as well as re-naming variables to distinguish them from non-standardized variables.

Usage

```
zt(x, fillfinite = 0, addz = T, saveattr = T)
```

Arguments

x	A data matrix for which the columns are to be standardized
fillfinite	The value to fill non-finite values with, by default zero.
addz	Boolean indicating whether letter 'z' should be appended to the variable names to indicate the standardization
saveattr	Boolean for if an 'attr' should be attached to the standardized vector, similar to how the R default function 'scale' conserves the centering and scaling values

Value

z-score standardized values (zero mean and unit variation), with non-finite values imputed by zero by default.

Examples

```
somedata <- cbind(rnorm(100), runif(100))
normdata <- zt(somedata)
head(normdata)
apply(normdata, MARGIN=2, FUN=mean)
apply(normdata, MARGIN=2, FUN=sd)
```

Index

∗ **datasets**
  DREAM, 7
  TYKS, 24
  TYKS_reduced, 24
  TYKSSIMU, 25

bootstrapRegCoefs, 3

coef,PSP-method *(*PSP-methods*)*, 19
conforminput, 4
cv, 4
cv.alpha, 5
cv.grid, 6

DREAM, 7

ePCR, 8

heatcv, 9

integrateRegCurve, 10
interact.all, 11
interact.part, 12

meanrank, 12

NelsonAalen, 13
normriskrank, 14

PEP-class, 15
PEP-methods, 16
plot *(*PSP-methods*)*, 19
plot,PSP,ANY-method *(*PSP-methods*)*, 19
plot,PSP-method *(*PSP-methods*)*, 19
predict,PEP-method *(*PEP-methods*)*, 16
predict,PSP-method *(*PSP-methods*)*, 19
print,PEP-method *(*PEP-methods*)*, 16
print,PSP-method *(*PSP-methods*)*, 19
PSP-class, 17
PSP-methods, 19
PSP.BOX *(*PSP-methods*)*, 19
PSP.BOX,PSP,ANY-method *(*PSP-methods*)*, 19
PSP.BOX,PSP-method *(*PSP-methods*)*, 19
PSP.CSP *(*PSP-methods*)*, 19
PSP.CSP,PSP,ANY-method *(*PSP-methods*)*, 19
PSP.CSP,PSP-method *(*PSP-methods*)*, 19
PSP.KM *(*PSP-methods*)*, 19
PSP.KM,PSP,ANY-method *(*PSP-methods*)*, 19
PSP.KM,PSP-method *(*PSP-methods*)*, 19
PSP.NA *(*PSP-methods*)*, 19
PSP.NA,PSP,ANY-method *(*PSP-methods*)*, 19
PSP.NA,PSP-method *(*PSP-methods*)*, 19
PSP.PCA *(*PSP-methods*)*, 19
PSP.PCA,PSP,ANY-method *(*PSP-methods*)*, 19
PSP.PCA,PSP-method *(*PSP-methods*)*, 19

score.cindex, 21
score.iAUC, 22

TimeSurvProb, 23
TYKS, 24
TYKS_reduced, 24
TYKSSIMU, 25

xMEDISIMU *(*TYKSSIMU*)*, 25
xTEXTSIMU *(*TYKSSIMU*)*, 25

yMEDISIMU *(*TYKSSIMU*)*, 25
yTEXTSIMU *(*TYKSSIMU*)*, 25

zt, 26