Package ‘SubgrpID’

March 4, 2017

Type Package

Title Patient Subgroup Identification for Clinical Drug Development

Version 0.11

Date 2017-02-26

Author Xin Huang, Yan Sun, Saptarshi Chatterjee and Paul Trow

Maintainer Xin Huang <xhuang.stats@gmail.com>

Description Function Wrapper contains four algorithms for developing threshold-based multivariate (prognostic/predictive) biomarker signatures via bootstrapping and aggregating of thresholds from trees, Monte-Carlo variations of the Adaptive Indexing method and Patient Rule Induction Method. Variable selection is automatically built-in to these algorithms. Final signatures are returned with interaction plots for predictive signatures. Cross-validation performance evaluation and testing dataset results are also output.

License GPL-3

Depends R (>= 2.1.0),AIM, survival, ggplot2, Matrix

Imports rpart, stats, glmnet

RoxygenNote 6.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2017-03-04 08:21:41

R topics documented:

`SubgrpID-package` *Patient subgroup identification for clinical drug development*

Description

Prognostic and predictive biomarker signature development for Exploratory Subgroup Identification in Randomized Clinical Trials

Details

Package:	SubgrpID
Type:	Package
Version:	0.10
Date:	2017-01-25
License:	GPL-3

Author(s)

Xin Huang, Yan Sun, Saptarshi Chatterjee and Paul Trow Maintainer: Xin Huang <xhuang.stats@gmail.com>

References

Huang X. et al. (2017) Patient subgroup identification for clinical drug development. *Statistics in Medicine*, doi: 10.1002/sim.7236.

Chen G. et al. (2015) A PRIM approach to predictive-signature development for patient stratification *Statistics in Medicine*, **34**, 317-342.

Examples

```
  ## Not run:
    data(Sepsis.train)
    data(Sepsis.test)

    yvar="survival"
    xvars=names(Sepsis.train)[2:12]
    trtvar="THERAPY"

    set.seed(123)
    subgrp <- SubgrpID(data.train=Sepsis.train,
                       yvar=yvar,
                       trtvar=trtvar,
                       trtref="active",
                       xvars=xvars,
                       type="b",
                       des.res = "smaller",
                       method="AIM.Rule")
    subgrp$res
    subgrp$train.stat
    subgrp$train.plot

## End(Not run)
```

---

aim.batting *The main AIM-BATTing function*

Description

This function finds the aim score for each subject in the dataset and using aim score as the predictor, performs BATTing to find the best threshold for each predictor.

Usage

```
aim.batting(y, x, censor.vec = NULL, trt.vec = NULL, trtref = NULL, type,
  n.boot, des.res = "larger", min.sigp.prcnt = 0.2, mc.iter = 1,
  mincut = 0.1, pre.filter = NULL, filter.method = NULL)
```

Arguments

`y`	data frame of the response variable.
`x`	data frame of predictors, each column of which corresponds to a variable.
`censor.vec`	data frame indicating censoring for survival data. For binary or continuous data, set censor.vec <- NULL.

`trt.vec`	data frame indicating whether or not the patient was treated. For the pronostic case, set trt.vec <- NULL.
`trtref`	code for treatment arm.
`type`	data type - "c" - continuous , "b" - binary, "s" - time to event - default = "c".
`n.boot`	number of bootstraps in bootstrapping step.
`des.res`	the desired response. "larger": prefer larger response; "smaller": prefer smaller response.
`min.sigp.prcnt`	desired proportion of signature positive group size.
`mc.iter`	# of iterations for the MC procedure to get a stable "best number of predictors".
`mincut`	the minimum cutting proportion for the binary rule at either end. It typically is between 0 and 0.2. It is the parameter in the functions of AIM package.
`pre.filter`	NULL, no prefiltering conducted;"opt", optimized number of predictors selected; An integer: min(opt, integer) of predictors selected.
`filter.method`	NULL, no prefiltering, "univariate", univariate filtering; "glmnet", glmnet filtering; "unicart", CART filtering (only for prognostic case).

Value

A list containing variables in signature and their thresholds.

`aim.batting.wrapper` *Wrapper function for cv.aim.batting to be passed to kfold.cv.*

Description

Wrapper function for cv.aim.batting to be passed to kfold.cv.

Usage

```
aim.batting.wrapper(data, args)
```

Arguments

`data`	data frame equal to cbind(y, x), where y and x are inputs to aim.batting.
`args`	list containing all other input arguments to aim.batting except for x and y.

Value

prediction rule as returned by aim.batting.

aim.rule.batting *The main AIM-Rule-BATTing function*

Description

This function first uses AIM to get the candidate rules and then applies Sequential BATTing to get the best rule(s).

Usage

```
aim.rule.batting(y, x, censor.vec = NULL, trt.vec = NULL, trtref = NULL,
  type, n.boot, des.res = "larger", min.sigp.prcnt = 0.2, mc.iter = 1,
  mincut = 0.1, pre.filter = NULL, filter.method = NULL)
```

Arguments

`y`	data frame of the response variable.
`x`	data frame of predictors, each column of which corresponds to a variable.
`censor.vec`	data frame indicating censoring for survival data. For binary or continuous data, set censor.vec <- NULL.
`trt.vec`	data frame indicating whether or not the patient was treated. For the pronostic case, set trt.vec <- NULL.
`trtref`	code for treatment arm.
`type`	data type - "c" - continuous , "b" - binary, "s" - time to event - default = "c".
`n.boot`	number of bootstraps in bootstrapping step.
`des.res`	the desired response. "larger": prefer larger response; "smaller": prefer smaller response.
`min.sigp.prcnt`	desired proportion of signature positive group size.
`mc.iter`	# of iterations for the MC procedure to get a stable "best number of predictors".
`mincut`	the minimum cutting proportion for the binary rule at either end. It typically is between 0 and 0.2. It is the parameter in the functions of AIM package.
`pre.filter`	NULL, no prefiltering conducted;"opt", optimized number of predictors selected; An integer: min(opt, integer) of predictors selected.
`filter.method`	NULL, no prefiltering, "univariate", univariate filtering; "glmnet", glmnet filtering; "unicart", CART filtering (only for prognostic case).

Value

A list of containing variables in signature and their thresholds.

`aim.rule.batting.wrapper` *Wrapper function for aim.rule.batting, to be passed to kfold.cv*

Description

Wrapper function for aim.rule.batting, to be passed to kfold.cv

Usage

```
aim.rule.batting.wrapper(data, args)
```

Arguments

`data`	data frame equal to cbind(y, x), where y and x are inputs to aim.rule.batting.
`args`	list containing all other input arguments to aim.rule.batting except for x and y.

Value

prediction rule returned by aim.rule.batting.

`aim.score.pred` *Find score of cutoff (returned by aim.find.cutoff.pred) for predictive case.*

Description

Find score of cutoff (returned by aim.find.cutoff.pred) for predictive case.

Usage

```
aim.score.pred(data, yvar, censorvar, trtvar, trtref, xvar, type, cutoff, nsubj,
  min.sigp.prcnt)
```

Arguments

`data`	data frame containing the response, covariate, treatment variable and censoring variable (only for time to event response).
`yvar`	response variable name.
`censorvar`	censoring variable name 1:event; 0: censor.
`trtvar`	treatment variable name.
`trtref`	code for treatment arm.
`xvar`	covariate variable name.
`type`	"c" continuous; "s" survival; "b" binary.

`cutoff`	cutpoint of interest.
`nsubj`	number of subjects.
`min.sigp.prcnt`	desired proportion of signature positive group size.

Value

AIM score for a single covariate-cutoff combination.

`aim.score.prog` *Find score of cutoff (returned by aim.find.cutoff.pred) for prognostic case.*

Description

Find score of cutoff (returned by aim.find.cutoff.pred) for prognostic case.

Usage

```
aim.score.prog(data, yvar, censorvar, xvar, type, cutoff, nsubj, min.sigp.prcnt)
```

Arguments

`data`	data frame containing the response, covariate, treatment variable and censoring variable (only for time to event response).
`yvar`	response variable name.
`censorvar`	censoring variable name 1:event; 0: censor.
`xvar`	covariate variable name.
`type`	"c" continuous; "s" survival; "b" binary.
`cutoff`	cutpoint of interest.
`nsubj`	number of subjects.
`min.sigp.prcnt`	desired proportion of signature positive group size.

Value

AIM score for a single covariate-cutoff combination.

`backfit.cox.interaction` *An internal function used in cox.interaction*

Description

An internal function used in cox.interaction

Usage

```
backfit.cox.interaction(x, trt, y, delta, cutp, mincut = 0)
```

Arguments

`x`	the predictor matrix.
`trt`	the treatment indicator vector.
`y`	the vector of the time to event response variable.
`delta`	status indicator: 1=failure 0=alive
`cutp`	a specific cutpoint
`mincut`	the minimum cutting proportion for the binary rule at either end. It typically is between 0 and 0.2. It is the parameter in the functions of AIM package.

`balanced.folds` *Create balanced folds for cross-validation.*

Description

Create balanced folds for cross-validation.

Usage

```
balanced.folds(y, nfolds = min(min(table(y)), 10))
```

Arguments

`y`	the response vector
`nfolds`	number of folds

Value

This function returns balanced folds

batting.pred *Main predictive BATTing function*

Description

Main predictive BATTing function

Usage

```
batting.pred(dataset, ids, yvar, censorvar, trtvar, type, class.wt, xvar,
  n.boot, des.res, min.sigp.prcnt)
```

Arguments

`dataset`	input dataset in data frame
`ids`	training indices
`yvar`	response variable name
`censorvar`	censoring variable name 1:event; 0: censor.
`trtvar`	treatment variable name
`type`	"c" continuous; "s" survival; "b" binary
`class.wt`	vector of length 2 used to weight the accuracy score , useful when there is class imbalance in binary data defaults to c(1,1)
`xvar`	name of predictor for which cutpoint needs to be obtained
`n.boot`	number of bootstraps for BATTing step.
`des.res`	the desired response. "larger": prefer larger response. "smaller": prefer smaller response.
`min.sigp.prcnt`	desired proportion of signature positive group size for a given cutoff.

Value

a signature rule consisting of variable name, direction, optimal cutpoint and the corresponding p-value.

`batting.prog` *Main prognostic BATTing function*

Description

Main prognostic BATTing function

Usage

```
batting.prog(dataset, ids, yvar, censorvar, type, class.wt, xvar, n.boot,
  des.res, min.sigp.prcnt)
```

Arguments

`dataset`	input dataset in data frame
`ids`	training indices
`yvar`	response variable name
`censorvar`	censoring variable name 1:event; 0: censor.
`type`	"c" continuous; "s" survival; "b" binary
`class.wt`	vector of length 2 used to weight the accuracy score , useful when there is class imbalance in binary data defaults to c(1,1)
`xvar`	name of predictor for which cutpoint needs to be obtained
`n.boot`	number of bootstraps for BATTing step.
`des.res`	the desired response. "larger": prefer larger response. "smaller": prefer smaller response.
`min.sigp.prcnt`	desired proportion of signature positive group size for a given cutoff.

Value

a signature rule consisting of variable name, direction, optimal cutpoint and the corresponding p-value.

`binary.stats` *A function for binary statistics*

Description

A function for binary statistics

Usage

```
binary.stats(pred.class, y.vec)
```

Arguments

`pred.class`	predicted output for each subject
`y.vec`	response vector

Value

a data frame with sensitivity, specificity, NPV, PPV and accuracy

---

`combine.condition` *Internal function*

Description

Internal function

Usage

```
combine.condition(trace.inside.condition, new.inside.condition)
```

Arguments

`trace.inside.condition`	list of signature rules
`new.inside.condition`	new signature rule

Value

updated list of signature rules

`cox.interaction` *Interaction Cox AIM*

Description

Interaction Cox AIM

Usage

```
cox.interaction(x, trt, y, delta, nsteps = 8, mincut = 0.1, backfit = F,
  maxnumcut = 1, dirp = 0)
```

Arguments

`x`	the predictor matrix.
`trt`	the treatment indicator vector.
`y`	the vector of the time to event response variable.
`delta`	status indicator: 1=failure 0=alive
`nsteps`	the maximum number of binary rules to be included in the index.
`mincut`	the minimum cutting proportion for the binary rule at either end. It typically is between 0 and 0.2. It is the parameter in the functions of AIM package.
`backfit`	a logical argument indicating whether the existing cutpoints are adjusted after including new binary rule.
`maxnumcut`	the maximum number of binary splits per predictor.
`dirp`	a vector for pre-specified direction of the binary split for each of the predictors. 0 represents "no pre-given direction"; 1 represents "(x>cut)"; -1 represents "(x<cut)". Alternatively, "dirp=0" represents that there is no pre-given direction for any of the predictor.

Value

"cox.interaction" returns "maxsc", which is the observed partial likelihood score test statistics for the index*treatment interaction in the fitted model and "res", which is a list with components

---

`create.training.dataset.index` *create training/testing dataset indexes.*

Description

create training/testing dataset indexes.

Usage

```
create.training.dataset.index(training.percent, n)
```

Arguments

`training.percent`	percentage of subjects in training data as mentioned in prim.train function.
`n`	number of sbjects in the whole dataset.

Value

a list containing training and test data indices.

`cv.aim.batting`	*The function for CV in aim.batting*

Description

Implements k-fold cross validation for aim.batting.

Usage

```
cv.aim.batting(y, x, censor.vec = NULL, trt.vec = NULL, trtref = NULL,
  type, n.boot, des.res = "larger", min.sigp.prcnt = 0.2, mc.iter = 1,
  mincut = 0.1, pre.filter = NULL, filter.method = NULL, k.fold = 5,
  cv.iter = 50, max.iter = 500)
```

Arguments

`y`	data frame containing the response
`x`	data frame containing the predictor
`censor.vec`	data frame giving the censor status (only for TTE data , censor=0,event=1) - default = NULL
`trt.vec`	data frame giving the censor status (only for TTE data , censor=0,event=1) - default = NULL
`trtref`	treatment reference indicator: 1=treatment, 0=control
`type`	data type - "c" - continuous , "b" - binary, "s" - time to event - default = "c"
`n.boot`	number of bootstraps in bootstrapping step.
`des.res`	the desired response. "larger": prefer larger response. "smaller": prefer smaller response
`min.sigp.prcnt`	desired proportion of signature positive group size for a given cutoff.
`mc.iter`	# of iterations for the MC procedure to get a stable "best number of predictors"
`mincut`	the minimum cutting proportion for the binary rule at either end. It typically is between 0 and 0.2. It is the parameter in the functions of AIM package.
`pre.filter`	NULL, no prefiltering conducted;"opt", optimized number of predictors selected; An integer: min(opt, integer) of predictors selected
`filter.method`	NULL, no prefiltering, "univariate", univaraite filtering; "glmnet", glmnet filtering
`k.fold`	# cross-validation folds
`cv.iter`	Algotithm terminates after cv.iter successful iterations of cross-validation
`max.iter`	total # iterations (including unsuccessful) allowed.

Value

"cv.aim.batting" returns a list with following entries:

`stats.summary`	Summary of performance statistics.
`pred.classes`	Data frame containing the predictive clases (TRUE/FALSE) for each iteration.
`folds`	Data frame containing the fold indices (index of the fold for each row) for each iteration.
`sig.list`	List of length cv.iter * k.fold containing the signature generated at each of the k folds, for all iterations.
`error.log`	List of any error messages that are returned at an iteration.
`interplot`	Treatment*subgroup interaction plot for predictive case

`cv.aim.rule.batting` *The function for CV in aim.rule.batting*

Description

Implements k-fold cross validation for aim.batting.

Usage

```
cv.aim.rule.batting(y, x, censor.vec = NULL, trt.vec = NULL,
  trtref = NULL, type, n.boot, des.res = "larger", min.sigp.prcnt = 0.2,
  mc.iter = 1, mincut = 0.1, pre.filter = NULL, filter.method = NULL,
  k.fold = 5, cv.iter = 50, max.iter = 500)
```

Arguments

`y`	data frame containing the response
`x`	data frame containing the predictor
`censor.vec`	data frame giving the censor status (only for TTE data , censor=0,event=1) - default = NULL
`trt.vec`	data frame giving the censor status (only for TTE data , censor=0,event=1) - default = NULL
`trtref`	treatment reference indicator: 1=treatment, 0=control
`type`	data type - "c" - continuous , "b" - binary, "s" - time to event - default = "c"
`n.boot`	number of bootstraps in bootstrapping step.
`des.res`	the desired response. "larger": prefer larger response. "smaller": prefer smaller response
`min.sigp.prcnt`	desired proportion of signature positive group size for a given cutoff.
`mc.iter`	# of iterations for the MC procedure to get a stable "best number of predictors"
`mincut`	the minimum cutting proportion for the binary rule at either end. It typically is between 0 and 0.2. It is the parameter in the functions of AIM package.

`pre.filter`	NULL, no prefiltering conducted;"opt", optimized number of predictors selected; An integer: min(opt, integer) of predictors selected
`filter.method`	NULL, no prefiltering, "univariate", univaraite filtering; "glmnet", glmnet filtering
`k.fold`	# cross-validation folds
`cv.iter`	Algotithm terminates after cv.iter successful iterations of cross-validation
`max.iter`	total # iterations (including unsuccessful) allowed.

Value

"cv.aim.batting" returns a list with following entries:

stats.summary: Summary of performance statistics

pred.classes: Data frame containing the predictive clases (TRUE/FALSE) for each iteration.

pred.classes: Data frame containing the predictive clases (TRUE/FALSE) for each iteration.

folds: Data frame containing the fold indices (index of the fold for each row) for each iteration

sig.list: List of length cv.iter * k.fold containing the signature generated at each of the k folds, for all iterations.

error.log: List of any error messages that are returned at an iteration.

`cv.cox.interaction` *A function for CV in Cox AIM with interaction.*

Description

A function for CV in Cox AIM with interaction.

Usage

```
cv.cox.interaction(x, trt, y, status, K.cv = 5, num.replicate = 1, nsteps,
  mincut = 0.1, backfit = F, maxnumcut = 1, dirp = 0)
```

Arguments

`x`	the predictor matrix.
`trt`	the treatment indicator vector.
`y`	the vector of the time to event response variable.
`status`	status indicator: 1=failure 0=alive
`K.cv`	number of folds for CV.
`num.replicate`	number of CV iterations.
`nsteps`	the maximum number of binary rules to be included in the index.
`mincut`	the minimum cutting proportion for the binary rule at either end. It typically is between 0 and 0.2. It is the parameter in the functions of AIM package.

`backfit`	a logical argument indicating whether the existing cutpoints are adjusted after including new binary rule.
`maxnumcut`	the maximum number of binary splits per predictor.
`dirp`	a vector for pre-specified direction of the binary split for each of the predictors. 0 represents "no pre-given direction"; 1 represents "(x>cut)"; -1 represents "(x<cut)". Alternatively, "dirp=0" represents that there is no pre-given direction for any of the predictor.

Value

returns optimal number of binary rules based on CV along with CV partial likelihood score test statistics, pre-validated partial likelihood score test statistics and prevalidated fits for individual observation.

`cv.cox.main` *A function for the number of binary rules in the main effect AIM with time to event outcome*

Description

A function for the number of binary rules in the main effect AIM with time to event outcome

Usage

```
cv.cox.main(x, y, status, K.cv = 5, num.replicate = 1, nsteps,
  mincut = 0.1, backfit = F, maxnumcut = 1, dirp = 0)
```

Arguments

`x`	the predictor matrix.
`y`	the vector of the time to event response variable.
`status`	a logical argument vector indicating status of a patient: 1=failure, 0=alive.
`K.cv`	number of folds for CV.
`num.replicate`	number of CV iterations.
`nsteps`	the maximum number of binary rules to be included in the index.
`mincut`	the minimum cutting proportion for the binary rule at either end. It typically is between 0 and 0.2. It is the parameter in the functions of AIM package.
`backfit`	a logical argument indicating whether the existing cutpoints are adjusted after including new binary rule.
`maxnumcut`	the maximum number of binary splits per predictor.
`dirp`	a vector for pre-specified direction of the binary split for each of the predictors. 0 represents "no pre-given direction"; 1 represents "(x>cut)"; -1 represents "(x<cut)". Alternatively, "dirp=0" represents that there is no pre-given direction for any of the predictor.

Value

returns optimal number of binary rules based on CV along with CV partial likelhood score test statistics for the main effect, pre-validated partial likelhood score test statistics and prevalidated fits for individual observation.

`cv.folds` *Cross-validation folds.*

Description

Cross-validation folds.

Usage

```
cv.folds(n, folds = 10)
```

Arguments

`n`	number of observations.
`folds`	number of folds.

Value

a list containing the observation numbers for each fold.

`cv.lm.interaction` *A function for CV in linear AIM with interaction.*

Description

A function for CV in linear AIM with interaction.

Usage

```
cv.lm.interaction(x, trt, y, K.cv = 5, num.replicate = 1, nsteps,
  mincut = 0.1, backfit = F, maxnumcut = 1, dirp = 0)
```

Arguments

`x`	the predictor matrix.
`trt`	the treatment indicator vector.
`y`	the vector of the continuous response variable.
`K.cv`	number of folds for CV.
`num.replicate`	number of CV iterations.
`nsteps`	the maximum number of binary rules to be included in the index.
`mincut`	the minimum cutting proportion for the binary rule at either end. It typically is between 0 and 0.2. It is the parameter in the functions of AIM package.
`backfit`	a logical argument indicating whether the existing cutpoints are adjusted after including new binary rule.
`maxnumcut`	the maximum number of binary splits per predictor.
`dirp`	a vector for pre-specified direction of the binary split for each of the predictors. 0 represents "no pre-given direction"; 1 represents "(x>cut)"; -1 represents "(x<cut)". Alternatively, "dirp=0" represents that there is no pre-given direction for any of the predictor.

Value

returns optimal number of binary rules based on CV along with CV score test statistics, prevalidated score test statistics and prevalidated fits for individual observation.

`cv.lm.main` *A function for the number of binary rules in the main effect AIM with continuous outcome*

Description

A function for the number of binary rules in the main effect AIM with continuous outcome

Usage

```
cv.lm.main(x, y, K.cv = 5, num.replicate = 1, nsteps, mincut = 0.1,
  backfit = F, maxnumcut = 1, dirp = 0)
```

Arguments

`x`	the predictor matrix.
`y`	the vector of the continuous response variable.
`K.cv`	number of folds for CV.
`num.replicate`	number of CV iterations.
`nsteps`	the maximum number of binary rules to be included in the index.

`mincut`	the minimum cutting proportion for the binary rule at either end. It typically is between 0 and 0.2. It is the parameter in the functions of AIM package.
`backfit`	a logical argument indicating whether the existing cutpoints are adjusted after including new binary rule.
`maxnumcut`	the maximum number of binary splits per predictor.
`dirp`	a vector for pre-specified direction of the binary split for each of the predictors. 0 represents "no pre-given direction"; 1 represents "(x>cut)"; -1 represents "(x<cut)". Alternatively, "dirp=0" represents that there is no pre-given direction for any of the predictor.

Value

returns optimal number of binary rules based on CV along with CV score test statistics for the main effect, pre-validated score test statistics and prevalidated fits for individual observation.

---

cv.logistic.interaction

A function for CV in logistic AIM with interaction.

Description

A function for CV in logistic AIM with interaction.

Usage

```
cv.logistic.interaction(x, trt, y, K.cv = 5, num.replicate = 1, nsteps,
  mincut = 0.1, backfit = F, maxnumcut = 1, dirp = 0, weight = 1)
```

Arguments

`x`	the predictor matrix.
`trt`	the treatment indicator vector.
`y`	the vector of the binary response variable.
`K.cv`	number of folds for CV.
`num.replicate`	number of CV iterations.
`nsteps`	the maximum number of binary rules to be included in the index.
`mincut`	the minimum cutting proportion for the binary rule at either end. It typically is between 0 and 0.2. It is the parameter in the functions of AIM package.
`backfit`	a logical argument indicating whether the existing cutpoints are adjusted after including new binary rule.
`maxnumcut`	the maximum number of binary splits per predictor.
`dirp`	a vector for pre-specified direction of the binary split for each of the predictors. 0 represents "no pre-given direction"; 1 represents "(x>cut)"; -1 represents "(x<cut)". Alternatively, "dirp=0" represents that there is no pre-given direction for any of the predictor.

`weight`	a positive value for the weight given to outcomes. "weight=0" means that all observations are equally weighted.

Value

returns optimal number of binary rules based on CV along with CV score test statistics and pre-validated score test statistics for the treatment*index interaction and prevalidated fits for individual observation.

`cv.logistic.main` *A function for the number of binary rules in the main effect AIM with binary outcome*

Description

A function for the number of binary rules in the main effect AIM with binary outcome

Usage

```
cv.logistic.main(x, y, K.cv = 5, num.replicate = 1, nsteps, mincut = 0.1,
  backfit = F, maxnumcut = 1, dirp = 0, weight = 1)
```

Arguments

`x`	the predictor matrix.
`y`	the vector of the binary response variable.
`K.cv`	number of folds for CV.
`num.replicate`	number of CV iterations.
`nsteps`	the maximum number of binary rules to be included in the index.
`mincut`	the minimum cutting proportion for the binary rule at either end. It typically is between 0 and 0.2. It is the parameter in the functions of AIM package.
`backfit`	a logical argument indicating whether the existing cutpoints are adjusted after including new binary rule.
`maxnumcut`	the maximum number of binary splits per predictor.
`dirp`	a vector for pre-specified direction of the binary split for each of the predictors. 0 represents "no pre-given direction"; 1 represents "(x>cut)"; -1 represents "(x<cut)". Alternatively, "dirp=0" represents that there is no pre-given direction for any of the predictor.
`weight`	a positive value for the weight given to outcomes. "weight=0" means that all observations are equally weighted.

Value

returns optimal number of binary rules based on CV along with CV score test statistics for the main effect, pre-validated score test statistics and prevalidated fits for individual observation.

cv.pval *p-value calculation for each iteration of cross validation.*

Description

p-value calculation for each iteration of cross validation.

Usage

```
cv.pval(yvar, censorvar = NULL, trtvar = NULL, data, type = "s")
```

Arguments

yvar	response variable name.
censorvar	censor-variable name.
trtvar	treatment variable name. For prognostic case trtvar=NULL.
data	dataset containing response and predicted output.
type	data type - "c" - continuous , "b" - binary, "s" - time to event - default = "c".

Value

p-value based on response and prediction vector for each iteration.

cv.seqlr.batting *Cross Validation for Sequential BATTing*

Description

Cross Validation for Sequential BATTing

Usage

```
cv.seqlr.batting(y, x, censor.vec = NULL, trt.vec = NULL, trtref = NULL,
  type = "c", n.boot = 50, des.res = "larger", class.wt = c(1, 1),
  min.sigp.prcnt = 0.2, pre.filter = NULL, filter.method = NULL,
  k.fold = 5, cv.iter = 50, max.iter = 500)
```

Arguments

`y`	data frame containing the response
`x`	data frame containing the predictors
`censor.vec`	vector giving the censor status (only for TTE data , censor=0,event=1) : default = NULL
`trt.vec`	vector containing values of treatment variable (for predictive signature). Set trt.vec to NULL for prognostic signature.
`trtref`	code for treatment arm.
`type`	data type. "c" - continuous , "b" - binary, "s" - time to event : default = "c".
`n.boot`	number of bootstraps in BATTing step.
`des.res`	the desired response. "larger": prefer larger response. "smaller": prefer smaller response
`class.wt`	vector of length 2 used to weight the accuracy score , useful when there is class imbalance in binary data defaults to c(1,1)
`min.sigp.prcnt`	desired proportion of signature positive group size for a given cutoff.
`pre.filter`	NULL, no prefiltering conducted;"opt", optimized number of predictors selected; An integer: min(opt, integer) of predictors selected.
`filter.method`	NULL, no prefiltering, "univariate", univaraite filtering; "glmnet", glmnet filtering, "unicart": univariate rpart filtering for prognostic case.
`k.fold`	number of folds for CV.
`cv.iter`	algorithm terminates after cv.iter successful iterations of cross-validation.
`max.iter`	total number of iterations allowed (including unsuccessful ones).

Value

a list containing with following entries:

`stats.summary`	Summary of performance statistics.
`pred.classes`	Data frame containing the predictive clases (TRUE/FALSE) for each iteration.
`folds`	Data frame containing the fold indices (index of the fold for each row) for each iteration.
`sig.list`	List of length cv.iter * k.fold containing the signature generated at each of the k folds, for all iterations.
`error.log`	List of any error messages that are returned at an iteration.
`interplot`	Treatment*subgroup interaction plot for predictive case

`evaluate.cv.results` *Cross-validation Performance Evaluation*

Description

Take the raw output of kfold.cv and calculate performance statistics for each iteration of the cross-validation.

Usage

```
evaluate.cv.results(cv.data, y, censor.vec, trt.vec, type)
```

Arguments

`cv.data`	output of prediction function from kfold.cv
`y`	data frame of the response variable from CV data.
`censor.vec`	data frame indicating censoring for survival data. For binary or continuous data, set censor.vec <- NULL.
`trt.vec`	data frame indicating whether or not the patient was treated. For the pronostic case, set trt.vec <- NULL.
`type`	data type - "c" - continuous , "b" - binary, "s" - time to event - default = "c"

Value

a list containing raw statistics and fold information

`evaluate.results` *Get statistics for a single set of predictions.*

Description

Get statistics for a single set of predictions.

Usage

```
evaluate.results(y, predict.data, censor.vec = NULL, trt.vec = NULL,
  trtref = NULL, type)
```

Arguments

`y`	data frame of the response variable.
`predict.data`	output of prediction function from kfold.cv.
`censor.vec`	data frame indicating censoring for survival data. For binary or continuous data, set censor.vec <- NULL.
`trt.vec`	data frame indicating whether or not the patient was treated. For the pronostic case, set trt.vec <- NULL.
`trtref`	treatment reference.
`type`	data type - "c" - continuous , "b" - binary, "s" - time to event - default = "c".

Value

a list containing p-value and group statistics.

`filter` *Filter function for predictive/prognostic biomarker candidates for signature development*

Description

Filter function for Prognostic and preditive biomarker signature development for Exploratory Subgroup Identification in Randomized Clinical Trials

Usage

```
filter(data,
type="c",
yvar,
xvars,
censorvar=NULL,
trtvar=NULL,
trtref=1,
n.boot=50,
cv.iter=20,
pre.filter=length(xvars),
filter.method=NULL)
```

Arguments

`data`	input data frame
`type`	type of response variable: "c" continuous; "s" survival; "b" binary
`yvar`	variable (column) name for response variable
`xvars`	vector of variable names for predictors (covariates)
`censorvar`	variable name for censoring (1: event; 0: censor), default = NULL

`trtvar`	variable name for treatment variable, default = NULL (prognostic signature)
`trtref`	coding (in the column of trtvar) for treatment arm, default = 1 (no use for prognostic signature)
`n.boot`	number of bootstrap for the BATTing procedure
`cv.iter`	Algotithm terminates after cv.iter successful iterations of cross-validation, or after max.iter total iterations, whichever occurs first
`pre.filter`	NULL (default), no prefiltering conducted;"opt", optimized number of predictors selected; An integer: min(opt, integer) of predictors selected
`filter.method`	NULL (default), no prefiltering; "univariate", univaraite filtering; "glmnet", glmnet filtering

Details

The function contains two algorithms for filtering high-dimentional multivariate (prognostic/predictive) biomarker candidates via univariate fitering (used p-values of group difference for prognostic case, p-values of interaction term for predictive case); LASSO/Elastic Net method. (Tian L. et al 2012)

Value

`var`	a vector of filter results of variable names

References

Tian L, Alizadeh A, Gentles A, Tibshirani R (2012) A Simple Method for Detecting Interactions between a Treatment and a Large Number of Covariates. J Am Stat Assoc. 2014 Oct; 109(508): 1517-1532.

Examples

```
## Not run:
data(Sepsis.train)

yvar="survival"
xvars=names(Sepsis.train)[2:12]
trtvar="THERAPY"
trtref="active"
set.seed(123)

filter.res <- filter(data=Sepsis.train,
type="b",
yvar=yvar,
xvars=xvars,
trtvar=trtvar,
trtref=trtref,
pre.filter=20,
filter.method="univariate")

filter.res

set.seed(123)
```

```
filter.res <- filter(data=Sepsis.train,
type="b",
yvar=yvar,
xvars=xvars,
trtvar=trtvar,
trtref=trtref,
pre.filter="opt",
filter.method="glmnet")

filter.res

## End(Not run)
```

filter.glmnet *Flitering using MC glmnet*

Description

Flitering using MC glmnet

Usage

```
filter.glmnet(data, type, yvar, xvars, censorvar, trtvar, trtref, n.boot = 50,
  cv.iter = 20, pre.filter = length(xvars))
```

Arguments

`data`	input data frame
`type`	"c" continuous; "s" survival; "b" binary
`yvar`	response variable name
`xvars`	covariates variable name
`censorvar`	censoring variable name 1:event; 0: censor.
`trtvar`	treatment variable name
`trtref`	code for treatment arm
`n.boot`	number of bootstrap for filtering
`cv.iter`	number of iterations required for MC glmnet filtering
`pre.filter`	NULL, no prefiltering conducted;"opt", optimized number of predictors selected; An integer: min(opt, integer) of predictors selected

Value

variables selected after glmnet filtering

filter.unicart *rpart filtering (only for prognostic case)*

Description

rpart filtering (only for prognostic case)

Usage

```
filter.unicart(data, type, yvar, xvars, censorvar, trtvar, trtref = 1,
  pre.filter = length(xvars))
```

Arguments

data	input data frame
type	"c" continuous; "s" survival; "b" binary
yvar	response variable name
xvars	covariates variable name
censorvar	censoring variable name 1:event; 0: censor.
trtvar	treatment variable name
trtref	code for treatment arm
pre.filter	NULL, no prefiltering conducted;"opt", optimized number of predictors selected; An integer: min(opt, integer) of predictors selected

Value

selected covariates after rpart filtering

filter.univariate *Univariate Filtering*

Description

Univariate Filtering

Usage

```
filter.univariate(data, type, yvar, xvars, censorvar, trtvar, trtref = 1,
  pre.filter = length(xvars))
```

Arguments

`data`	input data frame
`type`	"c" continuous; "s" survival; "b" binary
`yvar`	response variable name
`xvars`	covariates variable name
`censorvar`	censoring variable name 1:event; 0: censor.
`trtvar`	treatment variable name
`trtref`	code for treatment arm
`pre.filter`	NULL, no prefiltering conducted;"opt", optimized number of predictors selected; An integer: min(opt, integer) of predictors selected

Value

covariate names after univariate filtering.

`find.pred.stats` *Find predictive stats from response and prediction vector*

Description

Find predictive stats from response and prediction vector

Usage

```
find.pred.stats(data, yvar, trtvar, type, censorvar)
```

Arguments

`data`	data frame with response and prediction vector
`yvar`	response variable name
`trtvar`	treatment variable name
`type`	data type - "c" - continuous , "b" - binary, "s" - time to event - default = "c".
`censorvar`	censoring variable name

Value

a data frame of predictive statistics

find.prog.stats *Find prognostic stats from response and prediction vector*

Description

Find prognostic stats from response and prediction vector

Usage

```
find.prog.stats(data, yvar, type, censorvar)
```

Arguments

data	data frame with response and prediction vector
yvar	response variable name
type	data type - "c" - continuous , "b" - binary, "s" - time to event - default = "c".
censorvar	censoring variable name

Value

a data frame of predictive statistics

get.var.counts.aim *Get counts of signature variables from output of cv.aim.batting.*

Description

Get counts of signature variables from output of cv.aim.batting.

Usage

```
get.var.counts.aim(sig.list, xvars)
```

Arguments

sig.list	signature list output from cv.aim.batting.
xvars	predictor variable names.

Value

counts of signature variables.

`get.var.counts.seq` *Get signature variables from output of seqlr.batting.*

Description

Get signature variables from output of seqlr.batting.

Usage

```
get.var.counts.seq(sig.list, xvars)
```

Arguments

`sig.list`	signature list returned by seqlr.batting.
`xvars`	predictor variable names

Value

the variables included in signature rules returned by seqlr.batting

`interaction.plot` *A function for interaction plot*

Description

A function for interaction plot

Usage

```
interaction.plot(data.eval, type, main = "Interaction Plot",
  trt.lab = c("Trt.", "Ctrl."))
```

Arguments

`data.eval`	output of evaluate.results or summarize.cv.stats
`type`	data type - "c" - continuous , "b" - binary, "s" - time to event - default = "c".
`main`	title of the plot
`trt.lab`	treatment label

Value

A ggplot object.

kfold.cv *Perform k-fold cross-validation of a model.*

Description

Perform k-fold cross-validation of a model.

Usage

```
kfold.cv(data, model.Rfunc, model.Rfunc.args, predict.Rfunc, predict.Rfunc.args,
  k.fold = 5, cv.iter = 50, strata, max.iter = 500)
```

Arguments

`data`	the CV data
`model.Rfunc`	Name of the model function.
`model.Rfunc.args`	List of input arguments to model.Rfunc.
`predict.Rfunc`	Name of the prediction function, which takes the prediction rule returned by model.Rfunc along with any input data (not necessarily the input data to kfold.cv) and returns a TRUE-FALSE predictionvector specifying the positive and negative classes for the data.
`predict.Rfunc.args`	List containing input arguments to predict.Rfunc, except for data and predict.rule.
`k.fold`	Number of folds of the cross-validation.
`cv.iter`	Number of iterations of the cross-validation. If model.Rfunc returns an error at any of the k.fold calls, the current iteration is aborted. Iterations are repeated until cv.iter successful iterations have occurred.
`strata`	Stratification vector of length the number of rows of data, usually corresponding to the vector of events.
`max.iter`	Function stops after max.iter iterations even if cv.iter successful iterations have not occurred.

Value

List of length 2 with the following fields:

cv.data - List of length cv.iter. Entry i contains the output of predict.Rfunc at the ith iteration.

sig.list - list of length cv.iter * k.fold, whose entries are the prediction.rules (signatures) returned by model.Rfunc at each k.fold iteration.

`make.arg.list` *Create a list of variables corresponding to the arguments of the function func.name and assigns values.*

Description

Create a list of variables corresponding to the arguments of the function func.name and assigns values.

Usage

```
make.arg.list(func.name)
```

Arguments

`func.name`	function name

Value

list of variables corresponding to the arguments of the function

`one.dropping` *Perform dropping one time in predictive case.*

Description

Perform dropping one time in predictive case.

Usage

```
one.dropping(d.inside, d.outside, trace.inside.condition, yvar, censorvar,
  trtvar, g.str, l.str, type, des.res)
```

Arguments

`d.inside`	the dataset for subjects in consideration after pasting.
`d.outside`	the dataset for subjects outside consideration after pasting.
`trace.inside.condition`	list of signature rules used for d.inside.
`yvar`	the name for response variable.
`censorvar`	the name for censoring (1: event; 0: censor), default = NULL.
`trtvar`	0-1 coded vector for treatment variable.
`g.str`	">=".
`l.str`	"<=".

`type`	type of response variable: "c" continuous (default); "s" survival; "b" binary.
`des.res`	the desired response. "larger": prefer larger response (default) "smaller": prefer smaller response.

Value

a data frame enlisting the signature rules (after dropping) ordered by treatment p-values in each group defined by the rules.

`one.dropping.prog` *Perform dropping one time in prognostic case.*

Description

Perform dropping one time in prognostic case.

Usage

```
one.dropping.prog(d.inside, d.outside, trace.inside.condition, yvar, censorvar,
  g.str, l.str, type, des.res)
```

Arguments

`d.inside`	the dataset for subjects in consideration after pasting.
`d.outside`	the dataset for subjects outside consideration after pasting.
`trace.inside.condition`	list of signature rules used for d.inside.
`yvar`	the name for response variable.
`censorvar`	the name for censoring (1: event; 0: censor), default = NULL.
`g.str`	">=".
`l.str`	"<=".
`type`	type of response variable: "c" continuous (default); "s" survival; "b" binary.
`des.res`	the desired response. "larger": prefer larger response (default) "smaller": prefer smaller response.

Value

a data frame enlisting the signature rules (after dropping) ordered by main effect p-values in each group defined by the rules.

`one.pasting` *Perform pasting one time in predictive case.*

Description

Perform pasting one time in predictive case.

Usage

```
one.pasting(d.inside, d.outside, trace.inside.condition, alpha, yvar, censorvar,
  trtvar, g.str, l.str, type, des.res)
```

Arguments

`d.inside`	the dataset for subjects in consideration after peeling.
`d.outside`	the dataset for subjects outside consideration after peeling.
`trace.inside.condition`	list of signature rules used for d.inside.
`alpha`	a parameter controlling the number of subjects in consideration.
`yvar`	the name for response variable.
`censorvar`	the name for censoring (1: event; 0: censor), default = NULL.
`trtvar`	0-1 coded vector for treatment variable.
`g.str`	">=".
`l.str`	"<=".
`type`	type of response variable: "c" continuous (default); "s" survival; "b" binary.
`des.res`	the desired response. "larger": prefer larger response (default) "smaller": prefer smaller response.

Value

a data frame enlisting the signature rules (after pasting) ordered by treatment p-values in each group defined by the rules.

`one.pasting.prog` *Perform pasting one time in prognostic case.*

Description

Perform pasting one time in prognostic case.

Usage

```
one.pasting.prog(d.inside, d.outside, trace.inside.condition, alpha, yvar,
  censorvar, g.str, l.str, type, des.res)
```

Arguments

`d.inside`	the dataset for subjects in consideration after peeling.
`d.outside`	the dataset for subjects outside consideration after peeling.
`trace.inside.condition`	list of signature rules used for d.inside.
`alpha`	a parameter controlling the number of subjects in consideration.
`yvar`	the name for response variable.
`censorvar`	the name for censoring (1: event; 0: censor), default = NULL.
`g.str`	">=".
`l.str`	"<=".
`type`	type of response variable: "c" continuous (default); "s" survival; "b" binary.
`des.res`	the desired response. "larger": prefer larger response (default) "smaller": prefer smaller response.

Value

a data frame enlisting the signature rules (after pasting) ordered by main effect p-values in each group defined by the rules.

`one.peeling` *Perform peeling one time in predictive case.*

Description

Perform peeling one time in predictive case.

Usage

```
one.peeling(d.inside, d.outside, xvars, alpha, min.size.inside, yvar, censorvar,
  trtvar, g.str, l.str, type, des.res)
```

Arguments

`d.inside`	the dataset for subjects in consideration.
`d.outside`	the dataset for subjects outside consideration.
`xvars`	the vector of variable names for predictors (covariates).
`alpha`	a parameter controlling the number of patients in consideration
`min.size.inside`	desired number of subjects in signature positive group size for a given cutoff.
`yvar`	the name of response variable.
`censorvar`	the name of censoring variable (1: event; 0: censor), default = NULL).
`trtvar`	0-1 coded vector for treatment variable.
`g.str`	">=".
`l.str`	"<=".
`type`	type of response variable: "c" continuous (default); "s" survival; "b" binary.
`des.res`	the desired response. "larger": prefer larger response (default) "smaller": prefer smaller response.

Value

a data frame enlisting the signature rules (after peeling) ordered by treatment p-values in each group defined by the rules

---

`one.peeling.prog` *Perform peeling one time in prognostic case.*

Description

Perform peeling one time in prognostic case.

Usage

```
one.peeling.prog(d.inside, d.outside, xvars, alpha, min.size.inside, yvar,
  censorvar, g.str, l.str, type, des.res)
```

Arguments

`d.inside`	the dataset for subjects in consideration.
`d.outside`	the dataset for subjects outside consideration.
`xvars`	the vector of variable names for predictors (covariates).
`alpha`	a parameter controlling the number of patients in consideration
`min.size.inside`	desired number of subjects in signature positive group size for a given cutoff.
`yvar`	the name of response variable.

`censorvar`	the name of censoring variable (1: event; 0: censor), default = NULL).
`g.str`	">=".
`l.str`	"<=".
`type`	type of response variable: "c" continuous (default); "s" survival; "b" binary.
`des.res`	the desired response. "larger": prefer larger response (default) "smaller": prefer smaller response.

Value

a data frame enlisting the signature rules (after peeling) ordered by main effect p-values in each group defined by the rules.

`permute.rows` *Ramdomly permute the rows of a matrix.*

Description

Ramdomly permute the rows of a matrix.

Usage

```
permute.rows(A)
```

Arguments

`A`	a matrix for which its rows have to be permuted.

Value

the matrix with permuted rows.

`permute.vector` *Randomly permute the entries of a vector.*

Description

Randomly permute the entries of a vector.

Usage

```
permute.vector(x)
```

Arguments

`x`	the vector for which its entries have to be permuted

Value

the permuted vector

`pred.aim` *Make predictions for data given prediction rule in predict.rule.*

Description

Make predictions for data given prediction rule in predict.rule.

Usage

```
pred.aim(x, predict.rule)
```

Arguments

`x`	the predictor matrix.
`predict.rule`	prediction rule returned by seq.batting.

Value

The input data with an added column, a logical vector indicating the prediction for each row of data.

`pred.aim.cv` *Make predictions for data given prediction rule in predict.rule*

Description

Make predictions for data given prediction rule in predict.rule

Usage

```
pred.aim.cv(data, predict.rule, args)
```

Arguments

`data`	Data frame of form cbind(y, x), where y and x are inputs to cv.seq.batting.
`predict.rule`	Prediction rule returned by seq.batting.
`args`	list of the form list(xvar=xvar, yvar=yvar)

Value

The input data with an added column, a logical vector indicating the prediction for each row of data.

`pred.prim` *Prediction function for PRIM*

Description

Prediction function for PRIM

Usage

```
pred.prim(data, predict.rule)
```

Arguments

`data`	input data frame (only covariates)
`predict.rule`	signature rules returned by prim.train

Value

The input data with an added column, a logical vector indicating the prediction for each row of data.

`pred.prim.cv` *Prediction function for PRIM CV*

Description

Prediction function for PRIM CV

Usage

```
pred.prim.cv(data, predict.rule, args)
```

Arguments

`data`	input data frame
`predict.rule`	signature rules as returned by prim.train
`args`	list of the form list(yvar=yvar)

Value

The input data with an added column, a logical vector indicating the prediction for each row of data.

`pred.seqlr` *Prediction function for Sequential BATTing*

Description

Assign positive and negative groups based on predict.rule, the output of seqlr.batting.

Usage

```
pred.seqlr(x, predict.rule)
```

Arguments

`x`	input predictors matrix
`predict.rule`	Prediction rule returned by seqlr.batting.

Value

a logical vector indicating the prediction for each row of data.

`pred.seqlr.cv` *Prediction function for CV Sequential BATTing*

Description

Assign positive and negative groups for cross-validation data given prediction rule in predict.rule.

Usage

```
pred.seqlr.cv(data, predict.rule, args)
```

Arguments

`data`	input data frame
`predict.rule`	Prediction rule returned by seqlr.batting.
`args`	Prediction rule arguments

Value

a logical vector indicating the prediction for each row of data.

prim.cv *Cross-validation for PRIM*

Description

Cross-validation for PRIM

Usage

```
prim.cv(data, yvar, censorvar, trtvar, trtref = NULL, xvars, type, des.res,
  alpha = c(0.05, 0.06, 0.07, 0.08, 0.09, 0.1, 0.2, 0.3, 0.4, 0.5),
  min.sigp.prcnt = 0.2, training.percent = 0.5, n.boot = 0,
  pre.filter = NULL, filter.method = NULL, k.fold = 5, cv.iter = 50,
  max.iter = 500)
```

Arguments

`data`	the input data frame
`yvar`	name for response variable
`censorvar`	name for censoring (1: event; 0: censor), default = NULL
`trtvar`	name for treatment variable, default = NULL (prognostic signature)
`trtref`	coding (in the column of trtvar) for treatment arm
`xvars`	vector of variable names for predictors (covariates)
`type`	type of response variable: "c" continuous (default); "s" survival; "b" binary
`des.res`	the desired response. "larger": prefer larger response (default) "smaller": prefer smaller response
`alpha`	a parameter controlling the number of patients in consideration
`min.sigp.prcnt`	desired proportion of signature positive group size for a given cutoff.
`training.percent`	percentage of subjects in the initial training data
`n.boot`	number of bootstrap for the variable selection procedure for PRIM
`pre.filter`	NULL, no prefiltering conducted;"opt", optimized number of predictors selected; An integer: min(opt, integer) of predictors selected
`filter.method`	NULL, no prefiltering, "univariate", univaraite filtering; "glmnet", glmnet filtering, "unicart": univariate rpart filtering for prognostic case
`k.fold`	number of folds for CV.
`cv.iter`	Algorithm terminates after cv.iter successful iterations of cross-validation
`max.iter`	total number of iterations allowed (including unsuccessful ones)

Value

a list containing with following entries:

`stats.summary`	Summary of performance statistics.
`pred.classes`	Data frame containing the predictive clases (TRUE/FALSE) for each iteration.
`folds`	Data frame containing the fold indices (index of the fold for each row) for each iteration.
`sig.list`	List of length cv.iter * k.fold containing the signature generated at each of the k folds, for all iterations.
`error.log`	List of any error messages that are returned at an iteration.
`interplot`	Treatment*subgroup interaction plot for predictive case

`prim.train` *The main PRIM function*

Description

The main PRIM function

Usage

```
prim.train(data, yvar, censorvar, trtvar, trtref = NULL, xvars, type, des.res,
  alpha = c(0.05, 0.06, 0.07, 0.08, 0.09, 0.1, 0.2, 0.3, 0.4, 0.5),
  min.sigp.prcnt = 0.2, training.percent = 0.5, n.boot = 0,
  pre.filter = NULL, filter.method = NULL)
```

Arguments

`data`	the input data frame
`yvar`	name for response variable
`censorvar`	name for censoring (1: event; 0: censor), default = NULL
`trtvar`	name for treatment variable, default = NULL (prognostic signature)
`trtref`	coding (in the column of trtvar) for treatment arm
`xvars`	vector of variable names for predictors (covariates)
`type`	type of response variable: "c" continuous (default); "s" survival; "b" binary
`des.res`	the desired response. "larger": prefer larger response (default) "smaller": prefer smaller response
`alpha`	a parameter controlling the number of patients in consideration
`min.sigp.prcnt`	desired proportion of signature positive group size for a given cutoff.
`training.percent`	percentage of subjects in the initial training data
`n.boot`	number of bootstrap for the variable selection procedure for PRIM

`pre.filter`	NULL, no prefiltering conducted;"opt", optimized number of predictors selected; An integer: min(opt, integer) of predictors selected
`filter.method`	NULL, no prefiltering, "univariate", univaraite filtering; "glmnet", glmnet filtering, "unicart": univariate rpart filtering for prognostic case

Value

the final list of rules selected by PRIM.

`prim.train.pred.once` *Apply PRIM one time on the training data for a fixed value of alpha in predictive case*

Description

this function applies the prim procedure (peeling, pasting, and dropping operations) on training data one time.

Usage

```
prim.train.pred.once(data, yvar, censorvar, trtvar, xvars, type, des.res,
  alpha = 0.1, min.size.inside = 20, pidx.train.test)
```

Arguments

`data`	input data frame
`yvar`	name for response variable
`censorvar`	name for censoring (1: event; 0: censor), default = NULL
`trtvar`	0-1 coded vector for treatment variable
`xvars`	vector of variable names for predictors (covariates)
`type`	type of response variable: "c" continuous (default); "s" survival; "b" binary
`des.res`	the desired response. "larger": prefer larger response (default) "smaller": prefer smaller response
`alpha`	a parameter controlling the number of patients in consideration
`min.size.inside`	desired number of subjects in signature positive group size for a given cutoff.
`pidx.train.test`	training and test data index as obtained from create.training.dataset.index

Value

a list containing signature rules and test result based on the signatures.

`prim.train.prog.once` *Apply PRIM one time on the training data for a fixed value of alpha (in prognostic case)*

Description

this function applies the prim procedure (peeling, pasting, and dropping operations) on training data one time.

Usage

```
prim.train.prog.once(data, yvar, censorvar, xvars, type, des.res, alpha = 0.1,
  min.size.inside = 20, pidx.train.test)
```

Arguments

`data`	input data frame
`yvar`	name for response variable
`censorvar`	name for censoring (1: event; 0: censor), default = NULL
`xvars`	vector of variable names for predictors (covariates)
`type`	type of response variable: "c" continuous (default); "s" survival; "b" binary
`des.res`	the desired response. "larger": prefer larger response (default) "smaller": prefer smaller response
`alpha`	a parameter controlling the number of patients in consideration
`min.size.inside`	desired number of subjects in signature positive group size for a given cutoff.
`pidx.train.test`	training and test data index as obtained from create.training.dataset.index

Value

a list containing signature rules and test result based on the signatures.

`prim.train.wrapper` *Wrapper function for PRIM CV*

Description

Wrapper function for PRIM CV

Usage

```
prim.train.wrapper(data, args)
```

Arguments

`data`	input data frame
`args`	list containing all other input arguments to prim.train except for x and y.

Value

prediction rule as returned by prim.train

`pval.cal` *Calculate p-value for treatment in each subgroup in predictive case*

Description

Calculate p-value for treatment in each subgroup in predictive case

Usage

```
pval.cal(data, yvar, censorvar, trtvar, type, des.res)
```

Arguments

`data`	input data frame
`yvar`	name for response variable
`censorvar`	name for censoring (1: event; 0: censor), default = NULL
`trtvar`	name for treatment variable, default = NULL (prognostic signature)
`type`	type of response variable: "c" continuous (default); "s" survival; "b" binary
`des.res`	the desired response. "larger": prefer larger response (default) "smaller": prefer smaller response

Value

p-value for the treatment given the dataset

`pval.cal.prog` *Calculate p-value for treatment in each subgroup in prognostic case*

Description

Calculate p-value for treatment in each subgroup in prognostic case

Usage

```
pval.cal.prog(data, yvar, censorvar, grp.id, type, des.res)
```

Arguments

`data`	input data frame
`yvar`	name for response variable
`censorvar`	name for censoring (1: event; 0: censor), default = NULL
`grp.id`	subgroup id
`type`	type of response variable: "c" continuous (default); "s" survival; "b" binary
`des.res`	the desired response. "larger": prefer larger response (default) "smaller": prefer smaller response

Value

p-value for the main effect given the dataset

`query.data` *internal function used in seqlr.batting*

Description

internal function used in seqlr.batting

Usage

```
query.data(data, rule)
```

Arguments

`data`	the given dataset
`rule`	rule is a vector of the form [x-variable, direction, cutoff, p-value]

Value

a logical variable indicating whether rules are satisfied or not.

`query.from.condition` *An internal function inside one.pasting.*

Description

An internal function inside one.pasting.

Usage

```
query.from.condition(d, condition, g.str = ">=", l.str = "<=")
```

Arguments

`d`	dataset for subjects in consideration.
`condition`	signature rule in consideration.
`g.str`	">="
`l.str`	"<="

Value

a vector of logical arguments indicating whether the conditions can be satisfied for the subjects in d.

`resample` *Creates a permutation of given size.*

Description

Creates a permutation of given size.

Usage

```
resample(x, size, ...)
```

Arguments

`x`	the x vector.
`size`	resampling size.
`...`	optional argument.

Value

A resample of x is returned.

`Sepsis.test` *Sepsis Trial testing dataset*

Description

This is a simulated dataset based on a Phase III clinical trial compared a novel treatment to the standard of care (control) in patients with severe sepsis. The outcome of interest is a binary endpoint indicating subjects death after 28 days of treatment. Available markers include demographic and clinical covariates, i.e., age, time from first sepsis-organ fail to start drug, sum of baseline SOFA socres (cardiovascular, hematology, hepaticrenal, and respiration scores), number of baseline organ failures, pre-infusion apache-ii score, baseline GLASGOW coma scale score, baseline activity of daily living score; and laboratory markers, i.e., baseline local platelets, creatinine, serum IL-6 concentration, local bilirunbin.

Usage

```
Sepsis.test
```

Format

Dataset as a data frame

Source

http://multxpert.com

References

Lipkovich I, Dmitrienko A, Denne J, Enas G (2011) Subgroup identification based on differential effect search–a recursive partitioning method for establishing response to treatment in patient subpopulations. Stat Med 30:2601-2621. doi: 10.1002/sim.4289

`Sepsis.train` *Sepsis Trial training dataset*

Description

This is a simulated dataset based on a Phase III clinical trial compared a novel treatment to the standard of care (control) in patients with severe sepsis. The outcome of interest is a binary endpoint indicating subjects death after 28 days of treatment. Available markers include demographic and clinical covariates, i.e., age, time from first sepsis-organ fail to start drug, sum of baseline SOFA socres (cardiovascular, hematology, hepaticrenal, and respiration scores), number of baseline organ failures, pre-infusion apache-ii score, baseline GLASGOW coma scale score, baseline activity of daily living score; and laboratory markers, i.e., baseline local platelets, creatinine, serum IL-6 concentration, local bilirunbin.

Usage

```
Sepsis.train
```

Format

Dataset as a data frame

Source

http://multxpert.com

References

Lipkovich I, Dmitrienko A, Denne J, Enas G (2011) Subgroup identification based on differential effect search–a recursive partitioning method for establishing response to treatment in patient subpopulations. Stat Med 30:2601-2621. doi: 10.1002/sim.4289

`seqlr.batting` *Perform sequential BATTing method.*

Description

Perform sequential BATTing method.

Usage

```
seqlr.batting(y, x, censor.vec = NULL, trt.vec = NULL, trtref = NULL,
  type = "c", n.boot = 50, des.res = "larger", class.wt = c(1, 1),
  min.sigp.prcnt = 0.2, pre.filter = NULL, filter.method = NULL)
```

Arguments

`y`	data frame containing the response.
`x`	data frame containing the predictors.
`censor.vec`	vector containing the censor status (only for TTE data , censor=0,event=1) - default = NULL.
`trt.vec`	vector containing values of treatment variable (for predictive signature). Set trt.vec to NULL for prognostic signature.
`trtref`	code for treatment arm.
`type`	data type. "c" - continuous , "b" - binary, "s" - time to event : default = "c".
`n.boot`	number of bootstraps in BATTing step.
`des.res`	the desired response. "larger": prefer larger response. "smaller": prefer smaller response
`class.wt`	vector of length 2 used to weight the accuracy score , useful when there is class imbalance in binary data defaults to c(1,1)

`min.sigp.prcnt`	desired proportion of signature positive group size for a given cutoff.
`pre.filter`	NULL, no prefiltering conducted;"opt", optimized number of predictors selected; An integer: min(opt, integer) of predictors selected
`filter.method`	NULL, no prefiltering, "univariate", univaraite filtering; "glmnet", glmnet filtering, "unicart": univariate rpart filtering for prognostic case.

Value

it returns a list of signature rules consisting of variable names, directions, thresholds and the log-likelihood at each step the signatures are applied.

`seqlr.batting.wrapper` *Wrapper function for seqlr.batting, to be passed to kfold.cv.*

Description

Wrapper function for seqlr.batting, to be passed to kfold.cv.

Usage

```
seqlr.batting.wrapper(data, args)
```

Arguments

`data`	data frame equal to cbind(y, x, trt, censor), where y and x are inputs to seqlr.batting.
`args`	list containing all other input arguments to seq.batting except for x and y. Also contains xvars=names(x) and yvar=names(y).

Value

prediction rule returned by seqlr.batting.

`seqlr.find.cutoff.pred` *Find cutoff for predictive case.*

Description

Find cutoff for predictive case.

Usage

```
seqlr.find.cutoff.pred(data, yvar, censorvar, xvar, trtvar, type, class.wt, dir,
  nsubj, min.sigp.prcnt)
```

Arguments

`data`	input data frame.
`yvar`	response variable name.
`censorvar`	censoring variable name.
`xvar`	name of predictor for which cutpoint needs to be obtained.
`trtvar`	treatment variable name.
`type`	"c" continuous; "s" survival; "b" binary.
`class.wt`	vector of length 2 used to weight the accuracy score , useful when there is class imbalance in binary data defaults to c(1,1).
`dir`	direction of cut.
`nsubj`	number of subjects.
`min.sigp.prcnt`	desired proportion of signature positive group size for a given cutoff.

Value

the optimal score (p-value of subgroup*treatment interaction) for a predictor variable.

`seqlr.find.cutoff.prog` *Find cutoff for prognostic case.*

Description

Find cutoff for prognostic case.

Usage

```
seqlr.find.cutoff.prog(data, yvar, censorvar, xvar, type, class.wt, dir, nsubj,
  min.sigp.prcnt)
```

Arguments

`data`	input data frame.
`yvar`	response variable name.
`censorvar`	censoring variable name.
`xvar`	name of predictor for which cutpoint needs to be obtained.
`type`	"c" continuous; "s" survival; "b" binary.
`class.wt`	vector of length 2 used to weight the accuracy score , useful when there is class imbalance in binary data defaults to c(1,1).
`dir`	direction of cut.
`nsubj`	number of subjects.
`min.sigp.prcnt`	desired proportion of signature positive group size for a given cutoff.

Value

the optimal score (p-value of main effect) for a predictor variable.

`seqlr.score.pred` *Compute score of cutoff for predictive case*

Description

Compute score of cutoff for predictive case

Usage

```
seqlr.score.pred(data, yvar, censorvar, xvar, trtvar, cutoff, type, class.wt,
  dir, nsubj, min.sigp.prcnt)
```

Arguments

`data`	input data frame.
`yvar`	response variable name.
`censorvar`	censoring variable name.
`xvar`	name of predictor for which cutpoint needs to be obtained.
`trtvar`	treatment variable name.
`cutoff`	a specific cutpoint for which the score needs to be computed.
`type`	"c" continuous; "s" survival; "b" binary.
`class.wt`	vector of length 2 used to weight the accuracy score , useful when there is class imbalance in binary data defaults to c(1,1).
`dir`	direction of cut.
`nsubj`	number of subjects.
`min.sigp.prcnt`	desired proportion of signature positive group size for a given cutoff.

Value

score (p-value of treatment*subgroup interaction) for the given cutoff.

`seqlr.score.prog` *Compute score of cutoff for prognostic case*

Description

Compute score of cutoff for prognostic case

Usage

```
seqlr.score.prog(data, yvar, censorvar, xvar, cutoff, type, class.wt, dir,
  nsubj, min.sigp.prcnt)
```

Arguments

`data`	input data frame.
`yvar`	response variable name.
`censorvar`	censoring variable name.
`xvar`	name of predictor for which cutpoint needs to be obtained.
`cutoff`	a specific cutpoint for which the score needs to be computed.
`type`	"c" continuous; "s" survival; "b" binary.
`class.wt`	vector of length 2 used to weight the accuracy score , useful when there is class imbalance in binary data defaults to c(1,1).
`dir`	direction of cut.
`nsubj`	number of subjects.
`min.sigp.prcnt`	desired proportion of signature positive group size for a given cutoff.

Value

score (p-value of main effect) for the given cutoff.

`SubgrpID` *Exploratory Subgroup Identification main function*

Description

Prognostic and predictive biomarker signature development for Exploratory Subgroup Identification in Randomized Clinical Trials

Usage

```
SubgrpID(data.train, data.test=NULL,
         yvar,
         censorvar=NULL,
         trtvar=NULL,
         trtref=NULL,
         xvars,
         type="c",
         n.boot=ifelse(method=="PRIM",0,25),
         des.res="larger",
         min.sigp.prcnt=0.20,
         pre.filter=NULL,
         filter.method=NULL,
         k.fold=5,
         cv.iter=20,
         max.iter=500,
         mc.iter=20,
         method=c("AIM.Rule"),
         train.percent.prim=0.5,
         do.cv=FALSE,
         out.file=NULL,
         file.path="",
         plots=F)
```

Arguments

`data.train`	data frame for training dataset
`data.test`	data frame for testing dataset, default = NULL
`yvar`	variable (column) name for response variable
`censorvar`	variable name for censoring (1: event; 0: censor), default = NULL
`trtvar`	variable name for treatment variable, default = NULL (prognostic signature)
`trtref`	coding (in the column of trtvar) for treatment arm
`xvars`	vector of variable names for predictors (covariates)
`type`	type of response variable: "c" continuous; "s" survival; "b" binary
`n.boot`	number of bootstrap for batting procedure, or the variable selection procedure for PRIM; for PRIM, when n.boot=0, bootstrapping for variable selection is not conducted
`des.res`	the desired response. "larger": prefer larger response. "smaller": prefer smaller response
`min.sigp.prcnt`	desired proportion of signature positive group size for a given cutoff
`pre.filter`	NULL (default), no prefiltering conducted;"opt", optimized number of predictors selected; An integer: min(opt, integer) of predictors selected
`filter.method`	NULL (default), no prefiltering; "univariate", univaraite filtering; "glmnet", glmnet filtering; "unicart", univariate rpart filtering for prognostic case

`k.fold`	cross-validation folds
`cv.iter`	Algotithm terminates after cv.iter successful iterations of cross-validation, or after max.iter total iterations, whichever occurs first
`max.iter`	total iterations, whichever occurs first
`mc.iter`	number of iterations for the Monte Carlo procedure to get a stable "best number of predictors"
`method`	algorithms performed for subgroup identification, one of the ("AIM", "AIM.Rule", "Seq.BT", "PRIM")
`train.percent.prim`	percentage of the sub-training set used only by PRIM method; if train.percent.prim=1, all data will be used both for sub-training and sub-testing inside the PRIM
`do.cv`	whether to perform cross validation for performance evaluation. TRUE or FALSE (Default)
`out.file`	Name of output result files excluding method name. If NULL no output file would be saved
`file.path`	default: current working directory. When specifying a dir, use "/" at the end. e.g. "TEMP/"
`plots`	default: F. whether to save plots

Details

The function contains four algorithms for developing threshold-based multivariate (prognostic/predictive) biomarker signatures via resampled tree-based algorithms (Sequential BATTing), Monte-Carlo variations of the Adaptive Indexing method (AIM and AIM-Rule)and Patient Rule Induction Method. Variable selection is automatically built-in to these algorithms. Final signatures are returned with interaction plots for predictive signatures. Cross-validation performance evaluation and testing dataset results are also output.

Value

`res`	list of all results from the algorithm
`train.stat`	list of subgroup statistics on training dataset
`test.stat`	list of subgroup statistics on testing dataset
`cv.res`	list of all results from cross-validation on training dataset
`train.plot`	interaction plot for training dataset
`test.plot`	interaction plot for testing dataset

Author(s)

Xin Huang, Yan Sun, Saptarshi Chatterjee and Paul Trow

References

Huang X. et al. (2017) Patient subgroup identification for clinical drug development. *Statistics in Medicine*, doi: 10.1002/sim.7236.

Chen G. et al. (2015) A PRIM approach to predictive-signature development for patient stratification *Statistics in Medicine*, **34**, 317-342.

Examples

```
   ## Not run:
     data(Sepsis.train)
     data(Sepsis.test)

     yvar="survival"
     xvars=names(Sepsis.train)[2:12]
     trtvar="THERAPY"

     set.seed(123)
     subgrp <- SubgrpID(data.train=Sepsis.train,
                        data.test=Sepsis.test,
                        yvar=yvar,
                        trtvar=trtvar,
                        trtref="active",
                        xvars=xvars,
                        type="b",
                        des.res = "smaller",
                        method="AIM.Rule")
     subgrp$res
     subgrp$train.stat
     subgrp$test.stat
     subgrp$train.plot
     subgrp$test.plot

 ## End(Not run)
```

`summarize.cv.stats` *Calculate summary statistics from raw statistics returned by evaluate.cv.results.*

Description

Calculate summary statistics from raw statistics returned by evaluate.cv.results.

Usage

```
summarize.cv.stats(raw.stats, trtvar, type)
```

Arguments

`raw.stats`	raw statistics from evaluate.cv.results
`trtvar`	treatment variable name
`type`	data type - "c" - continuous , "b" - binary, "s" - time to event - default = "c"

Value

a list containing p-values, summary statistics and group statistics.

Index