

# Package: SubgrpID (via r-universe)

September 1, 2024

**Type** Package

**Title** Patient Subgroup Identification for Clinical Drug Development

**Version** 0.12

**Description** Implementation of Sequential BATTing (bootstrapping and aggregating of thresholds from trees) for developing threshold-based multivariate (prognostic/predictive) biomarker signatures. Variable selection is automatically built-in. Final signatures are returned with interaction plots for predictive signatures. Cross-validation performance evaluation and testing dataset results are also output. Detail algorithms are described in Huang et al (2017) <[doi:10.1002/sim.7236](https://doi.org/10.1002/sim.7236)>.

**License** GPL (>= 2)

**Encoding** UTF-8

**RoxygenNote** 7.2.3

**Imports** glmnet, MASS, rpart, survival, Matrix, ggplot2

**NeedsCompilation** no

**Author** Xin Huang [aut, cre, cph], Yan Sun [aut], Saptarshi Chatterjee [aut], Paul Trow [aut]

**Maintainer** Xin Huang <[xin.huang@abbvie.com](mailto:xin.huang@abbvie.com)>

**Repository** CRAN

**Date/Publication** 2024-02-03 12:20:10 UTC

## Contents

balanced.folds . . . . .	2
batting.pred . . . . .	3
batting.prog . . . . .	4
binary.stats . . . . .	5
cv.folds . . . . .	5
cv.pval . . . . .	6
cv.seqlr.batting . . . . .	7
data.gen . . . . .	8

evaluate.cv.results . . . . .	9
evaluate.results . . . . .	10
filter . . . . .	11
filter.glmnet . . . . .	12
filter.unicart . . . . .	13
filter.univariate . . . . .	14
find.pred.stats . . . . .	15
find.prog.stats . . . . .	16
get.var.counts.seq . . . . .	16
interaction.plot . . . . .	17
kfold.cv . . . . .	17
make.arg.list . . . . .	19
permute.rows . . . . .	19
permute.vector . . . . .	20
pred.seqlr . . . . .	20
pred.seqlr.cv . . . . .	21
query.data . . . . .	21
resample . . . . .	22
seqlr.batting . . . . .	22
seqlr.batting.wrapper . . . . .	23
seqlr.find.cutoff.pred . . . . .	24
seqlr.find.cutoff.prog . . . . .	25
seqlr.score.pred . . . . .	26
seqlr.score.prog . . . . .	27
SubgrpID . . . . .	28
summarize.cv.stats . . . . .	31

**Index** **32**

---

balanced.folds	<i>balanced.folds</i>
----------------	-----------------------

---

**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

**Details**

Create balanced folds for cross-validation.

**Value**

This function returns balanced folds

---

batting.pred	<i>batting.pred</i>
--------------	---------------------

---

**Description**

Main predictive BATTing function

**Usage**

```
batting.pred(
  dataset,
  ids,
  yvar,
  censorvar,
  trtvar,
  type,
  class.wt,
  xvar,
  n.boot,
  des.res,
  min.sig.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.sig.prcnt	desired proportion of signature positive group size for a given cutoff.

**Details**

Main predictive BATTing function

**Value**

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

---

batting.prog	<i>batting.prog</i>
--------------	---------------------

---

**Description**

Main prognostic BATTing function

**Usage**

```
batting.prog(
  dataset,
  ids,
  yvar,
  censorvar,
  type,
  class.wt,
  xvar,
  n.boot,
  des.res,
  min.sig.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.sig.prcnt	desired proportion of signature positive group size for a given cutoff.

**Details**

Main prognostic BATTing function

**Value**

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

---

binary.stats	<i>binary.stats</i>
--------------	---------------------

---

**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

**Details**

A function for binary statistics

**Value**

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

---

cv.folds	<i>cv.folds</i>
----------	-----------------

---

**Description**

Cross-validation folds.

**Usage**

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

**Arguments**

n	number of observations.
folds	number of folds.

**Details**

Cross-validation folds.

**Value**

a list containing the observation numbers for each fold.

---

*cv.pval**cv.pval*

---

**Description**

p-value calculation for each iteration of cross validation.

**Usage**

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

**Arguments**

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

**Details**

p-value calculation for each iteration of cross validation.

**Value**

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

---

cv.seqlr.batting      *cv.seqlr.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.sig.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.sig.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", univariate 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).

## Details

Cross Validation for Sequential BATTing

## Value

a list containing with following entries:

**stats.summary** Summary of performance statistics.

**pred.classes** Data frame containing the predictive classes (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

---

data.gen

*data.gen*

---

## Description

Function for simulated data generation

## Usage

```
data.gen(
  n,
  k,
  prevalence = sqrt(0.5),
  prog.eff = 1,
  sig2,
  y.sig2,
  rho,
  rhos.bt.real,
  a.constent
)
```



**Arguments**

n	Total sample size
k	Number of markers
prevalence	prevalence of predictive biomarkers with values above the cutoff
prog.eff	effect size <i>beta</i> for prognostic biomarker
sig2	standard deviation of each marker
y.sig2	Standard Deviation of the error term in the linear component
rho	rho*sig2 is the entries for covariance matrix between pairs of different k markers
rhos.bt.real	correlation between each prognostic and predictive markers
a.constant	a constant is set such that there is no overall treatment effect

**Details**

Function for simulated data generation

**Value**

A list of simulated clinical trial data with heterogeneous prognostic and predictive biomarkers

**Examples**

```
n <- 500
k <- 10
prevalence <- sqrt(0.5)
rho<-0.2
sig2 <- 2
rhos.bt.real <- c(0, rep(0.1, (k-3)))*sig2
y.sig2 <- 1
prog.eff <- 0.5
effect.size <- 1
a.constant <- effect.size/(2*(1-prevalence))
ObsData <- data.gen(n=n, k=k, prevalence=prevalence, prog.eff=prog.eff,
  sig2=sig2, y.sig2=y.sig2, rho=rho,
  rhos.bt.real=rhos.bt.real, a.constant=a.constant)
```

---

evaluate.cv.results    *evaluate.cv.results*

---

**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.
sensor.vec	data frame indicating censoring for survival data. For binary or continuous data, set sensor.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"

**Details**

Cross-validation Performance Evaluation

**Value**

a list containing raw statistics and fold information

---

evaluate.results	<i>evaluate.results</i>
------------------	-------------------------

---

**Description**

Get statistics for a single set of predictions.

**Usage**

```
evaluate.results(
  y,
  predict.data,
  sensor.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.
sensor.vec	data frame indicating censoring for survival data. For binary or continuous data, set sensor.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".

**Details**

Get statistics for a single set of predictions.

**Value**

a list containing p-value and group statistics.

---

filter

*filter*

---

**Description**

Filter function for Prognostic and predictive biomarker signature development for Exploratory Sub-group 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	Algorithm terminates after cv.iter successful iterations of cross-validation, or after max.iter total iterations, whichever occurs first

<code>pre.filter</code>	NULL (default), no prefiltering conducted;"opt", optimized number of predictors selected; An integer: min(opt, integer) of predictors selected
<code>filter.method</code>	NULL (default), no prefiltering; "univariate", univariate filtering; "glmnet", glmnet filtering

**Details**

Filter function for predictive/prognostic biomarker candidates for signature development

The function contains two algorithms for filtering high-dimensional multivariate (prognostic/predictive) biomarker candidates via univariate filtering (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**

```
# no run
```

---

<code>filter.glmnet</code>	<i>filter.glmnet</i>
----------------------------	----------------------

---

**Description**

Filtering 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
sensorvar	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

**Details**

Flitering using MC glmnet

**Value**

variables selected after glmnet filtering

---

filter.unicart	<i>filter.unicart</i>
----------------	-----------------------

---

**Description**

rpart filtering

**Usage**

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

**Arguments**

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

**Details**

rpart filtering (only for prognostic case)

**Value**

selected covariates after rpart filtering

---

`filter.univariate`      *filter.univariate*

---

**Description**

Univariate Filtering

**Usage**

```
filter.univariate(  
  data,  
  type,  
  yvar,  
  xvars,  
  sensorvar,  
  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
sensorvar	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

**Details**

Univariate Filtering

**Value**

covariate names after univariate filtering.

---

find.pred.stats	<i>find.pred.stats</i>
-----------------	------------------------

---

**Description**

Find predictive stats from response and prediction vector

**Usage**

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

**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".
sensorvar	censoring variable name

**Details**

Find predictive stats from response and prediction vector

**Value**

a data frame of predictive statistics

---

find.prog.stats	<i>find.prog.stats</i>
-----------------	------------------------

---

**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

**Details**

Find prognostic stats from response and prediction vector

**Value**

a data frame of predictive statistics

---

get.var.counts.seq	<i>get.var.counts.seq</i>
--------------------	---------------------------

---

**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      *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

**Details**

A function for interaction plot

**Value**

A ggplot object.

---

kfold.cv      *kfold.cv*

---

**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**

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

**Details**

Perform k-fold cross-validation of a model.

**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 `i`th 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	<i>make.arg.list</i>
---------------	----------------------

---

**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
-----------	---------------

**Details**

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

**Value**

list of variables corresponding to the arguments of the function

---

permute.rows	<i>permute.rows</i>
--------------	---------------------

---

**Description**

Randomly permute the rows of a matrix.

**Usage**

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

**Arguments**

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

**Details**

Randomly permute the rows of a matrix.

**Value**

the matrix with permuted rows.

---

permute.vector	<i>permute.vector</i>
----------------	-----------------------

---

**Description**

Randomly permute the entries of a vector.

**Usage**

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

**Arguments**

x                    the vector for which its entries have to be permuted

**Details**

Randomly permute the entries of a vector.

**Value**

the permuted vector

---

pred.seqlr	<i>pred.seqlr</i>
------------	-------------------

---

**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.

**Details**

Prediction function for Sequential BATTing

**Value**

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

---

```
pred.seqlr.cv      pred.seqlr.cv
```

---

**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

**Details**

Prediction function for CV Sequential BATTing

**Value**

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

---

```
query.data      query.data
```

---

**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]

**Details**

internal function used in seqlr.batting

**Value**

a logical variable indicating whether rules are satisfied or not.

---

resample	<i>resample</i>
----------	-----------------

---

**Description**

Creates a permutation of given size.

**Usage**

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

**Arguments**

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

**Details**

Creates a permutation of given size.

**Value**

A resample of x is returned.

---

seqlr.battng	<i>seqlr.battng</i>
--------------	---------------------

---

**Description**

Perform sequential BATTing method.

**Usage**

```
seqlr.battng(  
  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.
sensor.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.sig.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", univariate filtering; "glmnet", glmnet filtering, "unicart": univariate rpart filtering for prognostic case.

**Details**

Perform sequential BATTing method.

**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 *seqlr.batting.wrapper*

---

**Description**

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

**Usage**

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

**Arguments**

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

**Details**

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

**Value**

prediction rule returned by `seqlr.batting`.

---

`seqlr.find.cutoff.pred`

*seqlr.find.cutoff.pred*

---

**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.

### Details

Find cutoff for predictive case.

### Value

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

---

seqlr.find.cutoff.prog

*seqlr.find.cutoff.prog*

---

### 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.sig.prcnt desired proportion of signature positive group size for a given cutoff.

### Details

Find cutoff for prognostic case.

### Value

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

---

seqlr.score.pred	<i>seqlr.score.pred</i>
------------------	-------------------------

---

### Description

Compute score of cutoff for predictive case

### Usage

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

### Details

Compute score of cutoff for predictive case

### Value

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

---

seqlr.score.prog            *seqlr.score.prog*

---

### 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.

**Details**

Compute score of cutoff for prognostic case

**Value**

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

---

<code>SubgrpID</code>	<i>SubgrpID</i>
-----------------------	-----------------

---

**Description**

Exploratory Subgroup Identification main function

**Usage**

```

SubgrpID(
  data.train,
  data.test = NULL,
  yvar,
  censorvar = NULL,
  trtvar = NULL,
  trtref = NULL,
  xvars,
  type = "c",
  n.boot = 25,
  des.res = "larger",
  min.sigp.prcnt = 0.2,
  pre.filter = NULL,
  filter.method = NULL,
  k.fold = 5,
  cv.iter = 20,
  max.iter = 500,
  mc.iter = 20,
  method = c("Seq.BT"),
  do.cv = FALSE,
  out.file = NULL,
  file.path = "",
  plots = FALSE
)

```

**Arguments**

<code>data.train</code>	data frame for training dataset
<code>data.test</code>	data frame for testing dataset, default = NULL
<code>yvar</code>	variable (column) name for response variable
<code>ensorvar</code>	variable name for censoring (1: event; 0: censor), default = NULL
<code>trtvar</code>	variable name for treatment variable, default = NULL (prognostic signature)
<code>trtref</code>	coding (in the column of <code>trtvar</code> ) for treatment arm
<code>xvars</code>	vector of variable names for predictors (covariates)
<code>type</code>	type of response variable: "c" continuous; "s" survival; "b" binary
<code>n.boot</code>	number of bootstrap for batting procedure, or the variable selection procedure for PRIM; for PRIM, when <code>n.boot=0</code> , bootstrapping for variable selection is not conducted
<code>des.res</code>	the desired response. "larger": prefer larger response. "smaller": prefer smaller response
<code>min.sigp.prcnt</code>	desired proportion of signature positive group size for a given cutoff
<code>pre.filter</code>	NULL (default), no prefiltering conducted; "opt", optimized number of predictors selected; An integer: $\min(\text{opt}, \text{integer})$ of predictors selected
<code>filter.method</code>	NULL (default), no prefiltering; "univariate", univariate filtering; "glmnet", glmnet filtering; "unicart", univariate rpart filtering for prognostic case
<code>k.fold</code>	cross-validation folds
<code>cv.iter</code>	Algorithm terminates after <code>cv.iter</code> successful iterations of cross-validation, or after <code>max.iter</code> total iterations, whichever occurs first
<code>max.iter</code>	total iterations, whichever occurs first
<code>mc.iter</code>	number of iterations for the Monte Carlo procedure to get a stable "best number of predictors"
<code>method</code>	current version only supports sequential-BATting ("Seq.BT") for subgroup identification
<code>do.cv</code>	whether to perform cross validation for performance evaluation. TRUE or FALSE (Default)
<code>out.file</code>	Name of output result files excluding method name. If NULL no output file would be saved
<code>file.path</code>	default: current working directory. When specifying a dir, use "/" at the end. e.g. "TEMP/"
<code>plots</code>	default: FALSE. whether to save plots

**Details**

Function for SubgrpID

**Value**

A list with SubgrpID output

**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

**Examples**

```
# no run
n <- 40
k <- 5
prevalence <- sqrt(0.5)
rho<-0.2
sig2 <- 2
rhos.bt.real <- c(0, rep(0.1, (k-3)))*sig2
y.sig2 <- 1
yvar="y.binary"
xvars=paste("x", c(1:k), sep="")
trtvar="treatment"
prog.eff <- 0.5
effect.size <- 1
a.constant <- effect.size/(2*(1-prevalence))
set.seed(888)
ObsData <- data.gen(n=n, k=k, prevalence=prevalence, prog.eff=prog.eff,
  sig2=sig2, y.sig2=y.sig2, rho=rho,
  rhos.bt.real=rhos.bt.real, a.constant=a.constant)
TestData <- data.gen(n=n, k=k, prevalence=prevalence, prog.eff=prog.eff,
  sig2=sig2, y.sig2=y.sig2, rho=rho,
  rhos.bt.real=rhos.bt.real, a.constant=a.constant)
subgrp <- SubgrpID(data.train=ObsData$data,
  data.test=TestData$data,
  yvar=yvar,
  trtvar=trtvar,
  trtref="1",
  xvars=xvars,
  type="b",
  n.boot=5, # suggest n.boot > 25, depends on sample size
  des.res = "larger",
  # do.cv = TRUE,
  # cv.iter = 2, # uncomment to run CV
  method="Seq.BT")

subgrp$res
subgrp$train.stat
subgrp$test.stat
subgrp$train.plot
subgrp$test.plot
#subgrp$cv.res$stats.summary #CV estimates of all results
```

---

summarize.cv.stats	<i>summarize.cv.stats</i>
--------------------	---------------------------

---

**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"

**Details**

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

**Value**

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

# Index

balanced.folds, [2](#)  
batting.pred, [3](#)  
batting.prog, [4](#)  
binary.stats, [5](#)

cv.folds, [5](#)  
cv.pval, [6](#)  
cv.seqlr.batting, [7](#)

data.gen, [8](#)

evaluate.cv.results, [9](#)  
evaluate.results, [10](#)

filter, [11](#)  
filter.glmnet, [12](#)  
filter.unicart, [13](#)  
filter.univariate, [14](#)  
find.pred.stats, [15](#)  
find.prog.stats, [16](#)

get.var.counts.seq, [16](#)

interaction.plot, [17](#)

kfold.cv, [17](#)

make.arg.list, [19](#)

permute.rows, [19](#)  
permute.vector, [20](#)  
pred.seqlr, [20](#)  
pred.seqlr.cv, [21](#)

query.data, [21](#)

resample, [22](#)

seqlr.batting, [22](#)  
seqlr.batting.wrapper, [23](#)  
seqlr.find.cutoff.pred, [24](#)  
seqlr.find.cutoff.prog, [25](#)

seqlr.score.pred, [26](#)  
seqlr.score.prog, [27](#)  
SubgrpID, [28](#)  
summarize.cv.stats, [31](#)