Package ‘logicFS’

January 30, 2017

Title Identification of SNP Interactions
Version 1.44.0
Date 2013-09-12
Author Holger Schwender
Maintainer Holger Schwender <holger.schw@gmx.de>
Depends LogicReg, mcbiopi
Suggests genefilter, siggenes
Description Identification of interactions between binary variables using Logic
Regression. Can, e.g., be used to find interesting SNP interactions. Contains
also a bagging version of logic regression for classification.
License LGPL (>= 2)
biocViews SNP, Classification, Genetics
NeedsCompilation no

R topics documented:

  data.logicfs ...................................................... 2
getMatEval ......................................................... 2
logic.bagging ....................................................... 3
logic.oob ........................................................... 6
logic.pimp ........................................................... 7
logicFS .............................................................. 7
make.snp.dummy ..................................................... 11
mlogreg .............................................................. 12
plot.logicFS .......................................................... 13
predict.logicBagg .................................................. 15
predict.mlogreg .................................................... 16
print.logicFS ........................................................ 16
vim.approxPval ........................................................ 17
vim.chisq ............................................................. 18
vim.ebam ............................................................. 19
vim.input ............................................................ 21
vim.logicFS .......................................................... 23
vim.norm ............................................................. 24
vim.permSNP ........................................................ 26
vim.set ............................................................... 28

Index 31
**data.logicfs**  
*Example Data of logicFS*

**Description**

`data.logicfs` contains two objects: a simulated matrix `data.logicfs` of 400 observations (rows) and 15 variables (columns) and a vector `cl.logicfs` of length 400 containing the class labels of the observations.

Each variable is categorical with realizations 1, 2 and 3. The first 200 observations are cases, the remaining are controls. If one of the following expression is TRUE, then the corresponding observation is a case:

- SNP1 == 3
- SNP2 == 1 AND SNP4 == 3
- SNP3 == 3 AND SNP5 == 3 AND SNP6 == 1

where SNP1 is in the first column of `data.logicfs`, SNP2 in the second, and so on.

**See Also**

`logic.bagging`, `logicFS`

---

**getMatEval**  
*Evaluate Prime Implicants*

**Description**

Computes the values of prime implicants for observations for which the values of the variables composing the prime implicants are available.

**Usage**

`getMatEval(data, vec.primes, check = TRUE)`

**Arguments**

- `data` a data frame in which each row corresponds to an observation, and each column to a binary variable.
- `vec.primes` a character vector naming the prime implicants that should be evaluated. Each of the variables composing these prime implicants must be represented by one column of `data`.
- `check` should some checks be done before the evaluation is performed? It is highly recommended not to change the default `check = TRUE`.

**Value**

a matrix in which each row corresponds to an observation (the same observations in the same order as in `data`, and each column to one of the prime implicants.

**Author(s)**

Holger Schwender, `<holger.schwender@udo.edu>`
**logic.bagging**

**Bagged Logic Regression**

**Description**

A bagging and subsampling version of logic regression. Currently available for the classification, the linear regression, and the logistic regression approach of logreg. Additionally, an approach based on multinomial logistic regressions as implemented in mlogreg can be used if the response is categorical.

**Usage**

```r
## Default S3 method:
logic.bagging(x, y, B = 100, useN = TRUE, ntrees = 1, nleaves = 8,
              glm.if.1tree = FALSE, replace = TRUE, sub.frac = 0.632,
              anneal.control = logreg.anneal.control(), oob = TRUE,
              onlyRemove = FALSE, prob.case = 0.5, importance = TRUE,
              addMatImp = FALSE, fast = FALSE, rand = NULL, ...)
```

```r
## S3 method for class 'formula'
logic.bagging(formula, data, recdom = TRUE, ...)
```

**Arguments**

- **x**: a matrix consisting of 0’s and 1’s. Each column must correspond to a binary variable and each row to an observation. Missing values are not allowed.
- **y**: a numeric vector or a factor specifying the values of a response for all the observations represented in `x`, where no missing values are allowed in `y`. If a numeric vector, then `y` either contains the class labels (coded by 0 and 1) or the values of a continuous response depending on whether the classification or logistic regression approach of logic regression, or the linear regression approach, respectively, should be used. If the response is categorical, then `y` must be a factor naming the class labels of the observations.
- **B**: an integer specifying the number of iterations.
- **useN**: logical specifying if the number of correctly classified out-of-bag observations should be used in the computation of the importance measure. If FALSE, the proportion of correctly classified oob observations is used instead. Ignored if `importance = FALSE`.
- **ntrees**: an integer indicating how many trees should be used.
  - For a binary response: If `ntrees` is larger than 1, the logistic regression approach of logic regression will be used. If `ntrees` is 1, then by default the classification approach of logic regression will be used (see `glm.if.1tree`).
  - For a continuous response: A linear regression model with `ntrees` trees is fitted in each of the `B` iterations.
  - For a categorical response: `n.lev - 1` logic regression models with `ntrees` trees are fitted, where `n.lev` is the number of levels of the response (for details, see `mlogreg`).
- **nleaves**: a numeric value specifying the maximum number of leaves used in all trees combined. See the help page of the function `logreg` of the package LogicReg for details.
glm.if.1tree: if `ntrees` is 1 and `glm.if.1tree` is TRUE the logistic regression approach of logic regression is used instead of the classification approach. Ignored if `ntrees` is not 1 or the response is not binary.

replace: should sampling of the cases be done with replacement? If TRUE, a bootstrap sample of size `length(cl)` is drawn from the `length(cl)` observations in each of the B iterations. If FALSE, ceiling(sub.frac * length(cl)) of the observations are drawn without replacement in each iteration.

sub.frac: a proportion specifying the fraction of the observations that are used in each iteration to build a classification rule if `replace` = FALSE. Ignored if `replace` = TRUE.

anneal.control: a list containing the parameters for simulated annealing. See the help page of `logreg.anneal.control` in the `LogicReg` package.

oob: should the out-of-bag error rate (classification and logistic regression) or the out-of-bag root mean square prediction error (linear regression), respectively, be computed?

onlyRemove: should in the single tree case the multiple tree measure be used? If TRUE, the prime implicants are only removed from the trees when determining the importance in the single tree case. If FALSE, the original single tree measure is computed for each prime implicant, i.e. a prime implicant is not only removed from the trees in which it is contained, but also added to the trees that do not contain this interaction. Ignored in all other than the classification case.

prob.case: a numeric value between 0 and 1. If the outcome of the logistic regression, i.e. the class probability, for an observation is larger than prob.case, this observation will be classified as case (or 1).

importance: should the measure of importance be computed?

addMatImp: should the matrix containing the improvements due to the prime implicants in each of the iterations be added to the output? (For each of the prime implicants, the importance is computed by the average over the B improvements.) Must be set to TRUE, if standardized importances should be computed using `vim.norm`, or if permutation based importances should be computed using `vim.signperm`.

fast: should a greedy search (as implemented in `logreg`) be used instead of simulated annealing?

rand: numeric value. If specified, the random number generator will be set into a reproducible state.

formula: an object of class `formula` describing the model that should be fitted.

data: a data frame containing the variables in the model. Each row of data must correspond to an observation, and each column to a binary variable (coded by 0 and 1) or a factor (for details, see `recdom`) except for the column comprising the response, where no missing values are allowed in data. The response must be either binary (coded by 0 and 1), categorical or continuous. If continuous, a linear model is fitted in each of the B iterations of `logicFS`. If categorical, the column of data specifying the response must be a factor. In this case, multinomial logic regressions are performed as implemented in `mlogreg`. Otherwise, depending on `ntrees` (and `glm.if.1tree`) the classification or the logistic regression approach of logic regression is used.

recdom: a logical value or vector of length `ncol(data)` comprising whether a SNP should be transformed into two binary dummy variables coding for a recessive and a dominant effect. If recdom is TRUE (and a logical value), then all factors/variables with three levels will be coded by two dummy variables as described in `make.snp.dummy`. Each level of each of the other factors (also factors specifying a SNP that shows
only two genotypes) is coded by one indicator variable. If `recdom` is `FALSE` (and a logical value), each level of each factor is coded by an indicator variable. If `recdom` is a logical vector, all factors corresponding to an entry in `recdom` that is `TRUE` are assumed to be SNPs and transformed into two binary variables as described above. All variables corresponding to entries of `recdom` that are `TRUE` (no matter whether `recdom` is a vector or a value) must be coded either by the integers 1 (coding for the homozygous reference genotype), 2 (heterozygous), and 3 (homozygous variant), or alternatively by the number of minor alleles, i.e. 0, 1, and 2, where no mixing of the two coding schemes is allowed. Thus, it is not allowed that some SNPs are coded by 1, 2, and 3, and others are coded by 0, 1, and 2.

for the `formula` method, optional parameters to be passed to the low level function `logic.bagging.default`. Otherwise, ignored.

**Value**

`logic.bagging` returns an object of class `logicBagg` containing

- `logreg.model` a list containing the $B$ logic regression models,
- `inbagg` a list specifying the $B$ Bootstrap samples,
- `vim` an object of class `logicFS` (if `importance = TRUE`),
- `oob.error` the out-of-bag error (if `oob = TRUE`),
- ... further parameters of the logic regression.

**Author(s)**

Holger Schwender, <holger.schwender@udo.edu>

**References**


**See Also**

`predict.logicBagg`, `plot.logicBagg`, `logicFS`

**Examples**

```r
## Not run:
# Load data.
data(data.logicfs)

# For logic regression and hence logic.bagging, the variables must
# be binary. data.logicfs, however, contains categorical data
# with realizations 1, 2 and 3. Such data can be transformed
# into binary data by
bin.snps<-make.snp.dummy(data.logicfs)

# To speed up the search for the best logic regression models
# only a small number of iterations is used in simulated annealing.
```
my.anneal<-logreg.anneal.control(start=2,end=-2,iter=10000)

# Bagged logic regression is then performed by
bagg.out<-logic.bagging(bin.snps,cl.logicfs,ntree=20,nleaves=10,
                          rand=123,anneal.control=my.anneal)

# The output of logic.bagging can be printed
bagg.out

# By default, also the importances of the interactions are
# computed
bagg.out$vim

# and can be plotted.
plot(bagg.out)

# The original variable names are displayed in
plot(bagg.out,coded=FALSE)

# New observations (here we assume that these observations are
# in data.logicfs) are assigned to one of the classes by
predict(bagg.out,data.logicfs)

## End(Not run)

logic.oob

Prime Implicants

Description
Computes the out-of-bag error of the classification rule comprised by a logicBagg object.

Usage
logic.oob(log.out, prob.case = 0.5)

Arguments

log.out an object of class logicBagg, i.e. the output of logic.bagging.
prob.case a numeric value between 0 and 1. If the logic regression models are logistic
regression models, i.e. if in logic.bagging ntree is set to a value larger than
1, or glm.if.1tree is set to TRUE, then an observation will be classified as case
(or more exactly, as 1) if the class probability is larger than prob.case.

Value
The out-of-bag error estimate.

Author(s)
Holger Schwender, <holger.schwender@udo.edu>

See Also
logic.bagging
**logic.pimp**

**Prime Implicants**

**Description**
Determined the prime implicants contained in the logic regression models comprised in an object of class `logicBagg`.

**Usage**

```r
logic.pimp(log.out)
```

**Arguments**

- `log.out`: an object of class `logicBagg`, i.e., the output of `logic.bagging`.

**Details**
Since we are interested in all potentially interested interactions and not in a minimum set of them, `logic.pimp` and returns all prime implicants and not a minimum number of them.

**Value**
A list consisting of the prime implicants for each of the B logic regression models of `log.out`.

**Author(s)**
Holger Schwender, <holger.schwender@udo.edu>

**See Also**
`logic.bagging`, `logicFS`, `prime.implicants`

**logicFS**

**Feature Selection with Logic Regression**

**Description**
Identification of interesting interactions between binary variables using logic regression. Currently available for the classification, the linear regression and the logistic regression approach of `logreg` and for a multinomial logic regression as implemented in `mlogreg`.

**Usage**

```r
## Default S3 method:
logicFS(x, y, B = 100, useN = TRUE, ntrees = 1, nleaves = 8,
glm.if.1tree = FALSE, replace = TRUE, sub.frac = 0.632,
anneal.control = logreg.anneal.control(), onlyRemove = FALSE,
prob.case = 0.5, addMatImp = TRUE, fast = FALSE, rand = NULL, ...)

## S3 method for class 'formula'
logicFS(formula, data, recdom = TRUE, ...)
```

## Default S3 method:
```r
logicFS(x, y, B = 100, useN = TRUE, ntrees = 1, nleaves = 8,
glm.if.1tree = FALSE, replace = TRUE, sub.frac = 0.632,
anneal.control = logreg.anneal.control(), onlyRemove = FALSE,
prob.case = 0.5, addMatImp = TRUE, fast = FALSE, rand = NULL, ...)
```

## S3 method for class 'formula'
```r
logicFS(formula, data, recdom = TRUE, ...)
```
Arguments

x a matrix consisting of 0’s and 1’s. Each column must correspond to a binary variable and each row to an observation. Missing values are not allowed.

y a numeric vector or a factor specifying the values of a response for all the observations represented in x, where missing values are not allowed in y. If a numeric vector, then y either contains the class labels (coded by 0 and 1) or the values of a continuous response depending on whether the classification or logistic regression approach of logic regression, or the linear regression approach, respectively, should be used. If the response is categorical, then y must be a factor naming the class labels of the observations.

B an integer specifying the number of iterations.

useN logical specifying if the number of correctly classified out-of-bag observations should be used in the computation of the importance measure. If FALSE, the proportion of correctly classified oob observations is used instead.

ntrees an integer indicating how many trees should be used.

For a binary response: If ntrees is larger than 1, the logistic regression approach of logic regression will be used. If ntrees is 1, then by default the classification approach of logic regression will be used (see glm.if.1tree).

For a continuous response: A linear regression model with ntrees trees is fitted in each of the B iterations.

For a categorical response: n.lev – 1 logic regression models with ntrees trees are fitted, where n.lev is the number of levels of the response (for details, see mlogreg).

nleaves a numeric value specifying the maximum number of leaves used in all trees combined. For details, see the help page of the function logreg of the package LogicReg.

glm.if.1tree if ntrees is 1 and glm.if.1tree is TRUE the logistic regression approach of logic regression is used instead of the classification approach. Ignored if ntrees is not 1, or the response is not binary.

replace should sampling of the cases be done with replacement? If TRUE, a Bootstrap sample of size length(cl) is drawn from the length(cl) observations in each of the B iterations. If FALSE, ceiling(sub.frac * length(cl)) of the observations are drawn without replacement in each iteration.

sub.frac a proportion specifying the fraction of the observations that are used in each iteration to build a classification rule if replace = FALSE. Ignored if replace = TRUE.

anneal.control a list containing the parameters for simulated annealing. See the help of the function logreg.anneal.control in the LogicReg package.

onlyRemove should in the single tree case the multiple tree measure be used? If TRUE, the prime implicants are only removed from the trees when determining the importance in the single tree case. If FALSE, the original single tree measure is computed for each prime implicant, i.e. a prime implicant is not only removed from the trees in which it is contained, but also added to the trees that do not contain this interaction. Ignored in all other than the classification case.

prob.case a numeric value between 0 and 1. If the outcome of the logistic regression, i.e. the predicted probability, for an observation is larger than prob.case this observation will be classified as case (or 1).

addMatImp should the matrix containing the improvements due to the prime implicants in each of the iterations be added to the output? (For each of the prime implicants,
the importance is computed by the average over the B improvements.) Must be set to TRUE, if standardized importances should be computed using \texttt{vim.norm}, or if permutation based importances should be computed using \texttt{vim.signperm}.

\texttt{fast} should a greedy search (as implemented in \texttt{logreg}) be used instead of simulated annealing?

\texttt{rand} numeric value. If specified, the random number generator will be set into a reproducible state.

\texttt{formula} an object of class \texttt{formula} describing the model that should be fitted.

\texttt{data} a data frame containing the variables in the model. Each row of data must correspond to an observation, and each column to a binary variable (coded by 0 and 1) or a factor (for details, see \texttt{recdom}) except for the column comprising the response, where no missing values are allowed in data. The response must be either binary (coded by 0 and 1), categorical or continuous. If continuous, a linear model is fitted in each of the B iterations of \texttt{logicFS}. If categorical, the column of data specifying the response must be a factor. In this case, multinomial logic regressions are performed as implemented in \texttt{mlogreg}. Otherwise, depending on \texttt{ntrees} (and \texttt{glm.if.1tree}) the classification or the logistic regression approach of logic regression is used.

\texttt{recdom} a logical value or vector of length ncol(data) comprising whether a SNP should be transformed into two binary dummy variables coding for a recessive and a dominant effect. If \texttt{recdom} is TRUE (and a logical value), then all factors/variables with three levels will be coded by two dummy variables as described in \texttt{make.snp.dummy}. Each level of each of the other factors (also factors specifying a SNP that shows only two genotypes) is coded by one indicator variable. If \texttt{recdom} is FALSE (and a logical value), each level of each factor is coded by an indicator variable. If \texttt{recdom} is a logical vector, all factors corresponding to an entry in \texttt{recdom} that is TRUE are assumed to be SNPs and transformed into two binary variables as described above. All variables corresponding to entries of \texttt{recdom} that are TRUE (no matter whether \texttt{recdom} is a vector or a value) must be coded either by the integers 1 (coding for the homozygous reference genotype), 2 (heterozygous), and 3 (homozygous variant), or alternatively by the number of minor alleles, i.e. 0, 1, and 2, where no mixing of the two coding schemes is allowed. Thus, it is not allowed that some SNPs are coded by 1, 2, and 3, and others are coded by 0, 1, and 2.

... for the \texttt{formula} method, optional parameters to be passed to the low level function \texttt{logicFS.default}. Otherwise, ignored.

### Value

An object of class \texttt{logicFS} containing

- \texttt{primes} the prime implicants,
- \texttt{vim} the importance of the prime implicants,
- \texttt{prop} the proportion of logic regression models that contain the prime implicants,
- \texttt{type} the type of model (1: classification, 2: linear regression, 3: logistic regression),
- \texttt{param} further parameters (if \texttt{addInfo} = \texttt{TRUE}),
- \texttt{mat.imp} the matrix containing the improvements if \texttt{addMatImp} = \texttt{TRUE}, otherwise, NULL,
- \texttt{measure} the name of the used importance measure,
- \texttt{useN} the value of \texttt{useN},
- \texttt{threshold} NULL,
- \texttt{mu} NULL.
Author(s)
Holger Schwender, <holger.schwender@udo.edu>

References


See Also
plot.logicFS, logic.bagging

Examples
```r
## Not run:
# Load data.
data(data.logicfs)

# For logic regression and hence logic.fs, the variables must
# be binary. data.logicfs, however, contains categorical data
# with realizations 1, 2 and 3. Such data can be transformed
# into binary data by
bin.snps<-make.snp.dummy(data.logicfs)

# To speed up the search for the best logic regression models
# only a small number of iterations is used in simulated annealing.
my.anneal<-logreg.anneal.control(start=2,end=-2,iter=10000)

# Feature selection using logic regression is then done by
log.out<-logicFS(bin.snps,cl.logicfs,B=20,nleaves=10,
                 rand=123,anneal.control=my.anneal)

# The output of logic.fs can be printed
log.out

# One can specify another number of interactions that should be
# printed, here, e.g., 15.
print(log.out,topX=15)

# The variable importance can also be plotted.
plot(log.out)

# And the original variable names are displayed in
plot(log.out,coded=FALSE)

## End(Not run)
```
**make.snp.dummy**

**SNPs to Dummy Variables**

**Description**

Transforms SNPs into binary dummy variables.

**Usage**

```r
make.snp.dummy(data)
```

**Arguments**

- `data` - a matrix in which each column corresponds to a SNP and each row to an observation. The genotypes of all SNPs must be either coded by 1 (for the homozygous reference genotype), 2 (heterozygous), and 3 (homozygous variant) or by 0, 1, 2. It is not allowed that some SNPs following the 1, 2, 3 coding scheme and some SNPs the 0, 1, 2 coding. Missing values are allowed, but please note that neither `logic.bagging` nor `logicFS` can handle missing values so that the missing values need to be imputed (preferably before an application of `make.snp.dummy`).

**Details**

`make.snp.dummy` assumes that the homozygous dominant genotype is coded by 1, the heterozygous genotype by 2, and the homozygous recessive genotype by 3. Alternatively, the three genotypes can be coded by the number of minor alleles, i.e. by 0, 1, and 2. For each SNP, two dummy variables are generated:

- **SNP.1** At least one of the bases explaining the SNP are of the recessive type.
- **SNP.2** Both bases are of the recessive type.

**Value**

A matrix with 2*ncol(data) columns containing 2 dummy variables for each SNP.

**Note**

See the R package `scrime` for more general functions for recoding SNPs.

**Author(s)**

Holger Schwender, <holger.schwender@udo.edu>
mlogreg

Multinomial Logic Regression

Description

Performs a multinomial logic regression for a nominal response by fitting a logic regression model (with logit as link function) for each of the levels of the response except for the level with the smallest value which is used as reference category.

Usage

```r
## S3 method for class 'formula'
mlogreg(formula, data, recdom = TRUE, ...)

## Default S3 method:
mlogreg(x, y, ntrees = 1, nleaves = 8, anneal.control = logreg.anneal.control(), select = 1, rand = NA, ...)
```

Arguments

- `formula`: an object of class `formula` describing the model that should be fitted.
- `data`: a data frame containing the variables in the model. Each column of `data` must correspond to a binary variable (coded by 0 and 1) or a factor (for details on factors, see `recdom`) except for the column comprising the response, and each row to an observation. The response must be a categorical variable with less than 10 levels. This response can be either a factor or of type numeric or character.
- `recdom`: a logical value or vector of length `ncol(data)` comprising whether a SNP should be transformed into two binary dummy variables coding for a recessive and a dominant effect. If `TRUE` (logical value), then all factors (variables) with three levels will be coded by two dummy variables as described in `make.snp.dummy`. Each level of each of the other factors (also factors specifying a SNP that shows only two genotypes) is coded by one indicator variable. If `FALSE` (logical value), each level of each factor is coded by an indicator variable. If `recdom` is a logical vector, all levels of each factor corresponding to an entry in `recdom` that is `TRUE` are assumed to be SNPs and transformed into the two binary variables described above. Each variable that corresponds to an entry of `recdom` that is `TRUE` (no matter whether `recdom` is a vector or a value) must be coded by the integers 1 (coding for the homozygous reference genotype), 2 (heterozygous), and 3 (homozygous variant).
- `x`: a matrix consisting of 0’s and 1’s. Each column must correspond to a binary variable and each row to an observation.
- `y`: either a factor or a numeric or character vector specifying the values of the response. The length of `y` must be equal to the number of rows of `x`.
- `ntrees`: an integer indicating how many trees should be used in the logic regression models. For details, see `logreg` in the `LogicReg` package.
- `nleaves`: a numeric value specifying the maximum number of leaves used in all trees combined. See the help page of the function `logreg` in the `LogicReg` package for details.
plot.logicFS

anneal.control a list containing the parameters for simulated annealing. For details, see the help page of logreg.anneal.control in the LogicReg package.

select numeric value. Either 0 for a stepwise greedy selection (corresponds to select = 6 in logreg) or 1 for simulated annealing.

rand numeric value. If specified, the random number generator will be set into a reproducible state.

... for the formula method, optional parameters to be passed to the low level function mlogreg.default. Otherwise, ignored.

Value

An object of class mlogreg composed of

model a list containing the logic regression models,
data a matrix containing the binary predictors,
cl a vector comprising the class labels,
ntrees a numeric value naming the maximum number of trees used in the logic regressions,
nleaves a numeric value comprising the maximum number of leaves used in the logic regressions,
fast a logical value specifying whether the faster search algorithm, i.e. the greedy search, has been used.

Author(s)

Holger Schwender, <holger.schwender@udo.edu>

References


See Also

predict.mlogreg, logic.bagging, logicFS

plot.logicFS Variable Importance Plot

Description

Generates a dotchart of the importance of the most important interactions for an object of class logicFS or logicBagg.
Usage

## S3 method for class 'logicFS'
plot(x, topX = 15, cex = 0.9, pch = 16, col = 1, show.prop = FALSE,
     force.topX = FALSE, coded = TRUE, add.thres = TRUE, thres = NULL,
     include0 = TRUE, add.v0 = TRUE, v0.col = "grey50", main = NULL, ...)

## S3 method for class 'logicBagg'
plot(x, topX = 15, cex = 0.9, pch = 16, col = 1, show.prop = FALSE,
     force.topX = FALSE, include0 = TRUE, add.v0 = TRUE,
     v0.col = "grey50", main = NULL, ...)

Arguments

x
   an object of either class logicFS or logicBagg.

topX
   integer specifying how many interactions should be shown. If topX is larger
   than the number of interactions contained in x all the interactions are shown.
   For further information, see force.topX.

cex
   a numeric value specifying the relative size of the text and symbols.

pch
   specifies the used symbol. See the help of par for details.

col
   the color of the text and the symbols. See the help of par for how colors can be
   specified.

show.prop
   if TRUE the proportions of models that contain the interactions of interest are
   shown. If FALSE (default) the importances of the interactions are shown.

force.topX
   if TRUE exactly topX interactions are shown. If FALSE (default) all interactions
   up to the topXth most important one and all interactions having the same impor-
   tance as the topXth most important one are shown.

coded
   should the coded variable names be displayed? Might be useful if the actual
   variable names are pretty long. The coded variable name of the j-th variable is
   Xj.

add.thres
   should a vertical line marking the threshold for a prime implicant to be called
   important be drawn in the plot? If TRUE, this vertical line will be drawn at NULL.

thres
   non-negative numeric value specifying the threshold for a prime implicant to be
   called important. If NULL and add.thres = TRUE, the suggested threshold from
   x will be used.

include0
   should the x-axis include zero regardless whether the importances of the shown
   interactions are much higher than 0?

add.v0
   should a vertical line be drawn at x = 0? Ignored if include0 = FALSE and all
   importances are larger than zero.

v0.col
   the color of the vertical line at x = 0. See the help page of par for how colors
   can be specified.

main
   character string naming the title of the plot. If NULL, the name of the importance
   measure is used.

...
   Ignored.

Author(s)

Holger Schwender, <holger.schwender@udo.edu>
predict.logicBagg

See Also

logicFS, logic.bagging

predict.logicBagg  Predict Method for logicBagg objects

Description

Prediction for test data using an object of class logicBagg.

Usage

## S3 method for class 'logicBagg'
predict(object, newData, prob.case = 0.5,
    type = c("class", "prob"), ...)

Arguments

object  an object of class logicBagg.
newData a matrix or data frame containing new data. If omitted object$data, i.e., the original training data, are used. Each row of newData must correspond to a new observation. Each row of newData must contain the same variable as the corresponding column of the data matrix used in logic.bagging, i.e., x if the default method of logic.bagging has been used, or data without the column containing the response if the formula method has been used.
prob.case a numeric value between 0 and 1. A new observation will be classified as case (or more exactly, as 1) if the class probability, i.e., the average of the predicted probabilities of the models (if the logistic regression approach of logic regression has been used), or the percentage of votes for class 1 (if the classification approach of logic regression has been used) is larger than prob.case. Ignored if type = "prob" or the response is quantitative.
type character vector indicating the type of output. If "class", a numeric vector of zeros and ones containing the predicted classes of the observations (using the specification of prob.case) will be returned. If "prob", the class probabilities or percentages of votes for class 1, respectively, for all observations are returned. Ignored if the response is quantitative.
... Ignored.

Value

A numeric vector containing the predicted classes (if type = "class") or the class probabilities (if type = "prob") of the new observations if the classification or the logistic regression approach of logic regression is used. If the response is quantitative, the predicted value of the response for all observations in the test data set is returned.

Author(s)

Holger Schwender, <holger.schwender@udo.edu>

See Also

logic.bagging
predict.mlogreg  

**Predict Method for mlogreg Objects**

**Description**

Prediction for test data using an object of class mlogreg.

**Usage**

```r
## S3 method for class 'mlogreg'
predict(object, newData, type = c("class", "prob"), ...)
```

**Arguments**

- `object` an object of class mlogreg, i.e.\ the output of the function mlogreg.
- `newData` a matrix or data frame containing new data. If omitted object$data, i.e.\ the original training data, are used. Each row of newData must correspond to a new observation. Each row of newData must contain the same variable as the corresponding column of the data matrix used in mlogreg, i.e.\ x if the default method of mlogreg has been used, or data without the column containing the response if the formula method has been used.
- `type` character vector indicating the type of output. If "class", a vector containing the predicted classes of the observations will be returned. If "prob", the class probabilities for each level and all observations are returned.
- `...` Ignored.

**Value**

A numeric vector containing the predicted classes (if type = "class"), or a matrix composed of the class probabilities (if type = "prob").

**Author(s)**

Holger Schwender, <holger.schwender@udo.edu>

**See Also**

mlogreg
vim.approxPval

Arguments

x an object of either class logicFS.
topX integer indicating how many interactions should be shown. Additionally to the topX most important interactions, any interaction having the same importance as the topX most important one are also shown.
show.prop should the proportions of models containing the interactions of interest also be shown?
coded should the coded variable names be displayed? Might be useful if the actual variable names are pretty long. The coded variable name of the j-th variable is Xj.
digits number of digits used in the output.
... Ignored.

Author(s)

Holger Schwender, <holger.schwender@udo.edu>

See Also

logicFS, vim.logicFS

vim.approxPval

Approximate P-Value Based Importance Measure

Description

Computes the importances based on an approximation to a t- or F-distribution.

Usage

vim.approxPval(object, version = 1, adjust = "bonferroni")

Arguments

object an object of class logicFS which contains the values of standardized importances. Only in the linear regression case, the importances in object are allowed to be non-standardized.
version either 1 or 2. If 1, then the importance measure is computed by 1 - padj, where padj is the adjusted p-value. If 2, the importance measure is determined by -log10(padj), where a raw p-value equal to 0 is set to 1/(10 * n.perm) to avoid infinitive importances.
adjust character vector naming the method with which the raw permutation based p-values are adjusted for multiplicity. If "qvalue", the function qvalue.cal from the package siggenes is used to compute q-values. Otherwise, p.adjust is used to adjust for multiple comparisons. See p.adjust for all other possible specifications of adjust. If "none", the raw p-values will be used.
An object of class `logicFS` containing the same object as `object` except for

`vim` the values of the importance measure based on an approximation to the t- or F-distribution,

`measure` the name of the used importance measure,

`threshold` 0.95 if `version = 1`, and \(-\log_{10}(0.05)\) if `version = 2`.

Author(s)

Holger Schwender, <holger.schwender@udo.edu>

References


See Also

`logic.bagging`, `logicFS`, `vim.input`, `vim.set`, `vim.permSet`

---

Determining the importance of interactions found by `logic.bagging` or `logicFS` by Pearson’s ChiSquare Statistic. Only available for the classification and the logistic regression approach of `logic regression`.

Usage

`vim.chisq(object, data = NULL, cl = NULL)`

Arguments

- `object` either an object of class `logicFS` or the output of an application of `logic.bagging` with `importance = TRUE`.
- `data` a data frame or matrix consisting of 0’s and 1’s in which each column corresponds to one of the explanatory variables used in the original analysis with `logic.bagging` or `logicFS`, and each row corresponds to an observation. Must be specified if `object` is an object of class `logicFS`, or `cl` is specified. If `object` is an object of class `logicBagg` and neither `data` nor `cl` is specified, `data` and `cl` stored in `object` is used to compute the ChiSquare statistics. It is, however, highly recommended to use new data to test the interactions contained in `object`, as they have been found using the `data` stored in `object`, and it is very likely that most of them will show up as interesting if they are tested on the same `data` set.
- `cl` a numeric vector of 0’s and 1’s specifying the class labels of the observations in `data`. Must be specified either if `object` is an object of class `logicFS`, or if `data` is specified.
Details

Currently Pearson’s ChiSquare statistic is computed without continuity correction. Contrary to `vim.logicFS` (and `vim.norm` and `vim.signperm`), `vim.chisq` does neither take the logic regression models into account nor uses the out-of-bag observations for computing the importances of the identified interactions. It “just” tests each of the found interactions on the whole data set by calculating Pearson’s ChiSquare statistic for each of these interactions. It is, therefore, highly recommended to use an independent data set for specifying the importances of these interactions with `vim.chisq`.

Value

An object of class `logicFS` containing

<table>
<thead>
<tr>
<th>primes</th>
<th>the prime implicants</th>
</tr>
</thead>
<tbody>
<tr>
<td>vim</td>
<td>the values of Pearson’s ChiSquare statistic,</td>
</tr>
<tr>
<td>prop</td>
<td>NULL,</td>
</tr>
<tr>
<td>type</td>
<td>NULL,</td>
</tr>
<tr>
<td>param</td>
<td>further parameters (if object is the output of <code>logicFS</code> or <code>vim.logicFS</code> with <code>addInfo = TRUE</code>),</td>
</tr>
<tr>
<td>mat.imp</td>
<td>NULL,</td>
</tr>
<tr>
<td>measure</td>
<td>&quot;ChiSquare Based&quot;,</td>
</tr>
<tr>
<td>threshold</td>
<td>the 1 - 0.05/m quantile of the ChiSquare distribution with one degree of freedom,</td>
</tr>
<tr>
<td>mu</td>
<td>NULL.</td>
</tr>
</tbody>
</table>

Author(s)

Holger Schwender, <holger.schwender@udo.edu>

See Also

`logic.bagging`, `logicFS`, `vim.logicFS`, `vim.norm`, `vim.ebam`

---

### vim.ebam

**EBAM Based Importance**

#### Description

Determines the importance of interactions found by `logic.bagging` or `logicFS` by an Empirical Bayes Analysis of Microarrays (EBAM). Only available for the classification and the logistic regression approach of logic regression.

#### Usage

```r
vim.ebam(object, data = NULL, cl = NULL, storeEBAM = FALSE, ...)
```
Arguments

- **object**: either an object of class `logicFS` or the output of an application of `logic.bagging` with `importance = TRUE`.

- **data**: a data frame or matrix consisting of 0’s and 1’s in which each column corresponds to one of the explanatory variables used in the original analysis with `logic.bagging` or `logicFS`, and each row corresponds to an observation. Must be specified if `object` is an object of class `logicFS`, or `cl` is specified. If `object` is an object of class `logicBagg` and neither `data` nor `cl` is specified, `data` and `cl` stored in `object` is used to compute the ChiSquare statistics. It is, however, highly recommended to use new `data` to test the interactions contained in `object`, as they have been found using the `data` stored in `object`, and it is very likely that most of them will show up as interesting if they are tested on the same data set.

- **cl**: a numeric vector of 0’s and 1’s specifying the class labels of the observations in `data`. Must be specified either if `object` is an object of class `logicFS`, or if `data` is specified.

- **storeEBAM**: logical specifying whether the output of the EBAM analysis should be stored in the output of `vim.ebam`.

- **...**: further arguments of `ebam` and `cat.ebam`. For details, see the help files of these functions from the package `siggenes`.

Details

For each interaction found by `logic.bagging` or `logicFS`, the posterior probability that this interaction is significant is computed using the Empirical Bayes Analysis of Microarrays (EBAM). These posterior probabilities are used as the EBAM based importances of the interactions.

The test statistic underlying this EBAM analysis is Pearson’s ChiSquare statistic. Currently, the value of this statistic is computed without continuity correction.

Contrary to `vim.logicFS` (and `vim.norm` and `vim.signperm`), `vim.ebam` does neither take the logic regression models into account nor uses the out-of-bag observations for computing the importances of the identified interactions. It “just” tests each of the found interactions on the whole data set by calculating Pearson’s ChiSquare statistic for each of these interactions and performing an EBAM analysis. It is, therefore, highly recommended to use an independent data set for specifying the importances of these interactions with `vim.ebam`.

Value

An object of class `logicFS` containing

- **primes**: the prime implicants,
- **vim**: the posterior probabilities of the interactions,
- **prop**: `NULL`,
- **type**: `NULL`,
- **param**: further parameters (if `object` is the output of `logicFS` or `vim.logicFS` with `addInfo = TRUE`),
- **mat.imp**: `NULL`,
- **measure**: “EBAM Based”;
- **threshold**: the value of `delta` used in the EBAM analysis (see help files for `ebam`); by default: 0.9,
vim.input

mu NULL,

ebam an object of class EBAM (only available if storeEBAM = TRUE).

Author(s)

Holger Schwender, <holger.schwender@udo.edu>

References


See Also

logic.bagging, logicFS, vim.logicFS, vim.norm, vim.chisq

vim.input  VIM for Inputs

Description

Quantifies the importance of each input variable occurring in at least one of the logic regression models found in the application of logic.bagging.

Usage

```r
vim.input(object, useN = NULL, iter = NULL, prop = TRUE,
standardize = NULL, mu = 0, addMatImp = FALSE,
prob.case = 0.5, rand = NA)
```

Arguments

- **object**: an object of class `logicBagg`, i.e., the output of `logic.bagging`
- **useN**: logical specifying if the number of correctly classified out-of-bag observations should be used in the computation of the importance measure. If `FALSE`, the proportion of correctly classified oob observations is used instead. If `NULL` (default), then the specification of `useN` in `object` is used.
- **iter**: integer specifying the number of times the values of the considered variable are permuted in the computation of its importance. If `NULL` (default), the values of the variable are not permuted, but the variable is removed from the model.
- **prop**: should the proportion of logic regression models containing the respective variable also be computed?
- **standardize**: should a standardized version of the importance measure for a set of variables be returned? By default, `standardize = TRUE` is used in the classification and the (multinomial) logistic regression case, and `standardize` is set to `FALSE` in the linear regression case. For details, see `mu`.
- **mu**: a non-negative numeric value. Ignored if `standardize = FALSE`. Otherwise, a t-statistic for testing the null hypothesis that the importance of the respective variable is equal to `mu` is computed.
addMatImp should the matrix containing the improvements due to each of the variables in each of the logic regression models be added to the output?

prob.case a numeric value between 0 and 1. If the logistic regression approach of logic regression has been used in logic.bagging, then an observation will be classified as a case (or more exactly, as 1), if the class probability of this observation is larger than prob.case. Otherwise, prob.case is ignored.

rand an integer for setting the random number generator in a reproducible case.

Value

An object of class logicFS containing

vim the importances of the variables,
prop the proportion of logic regression models containing the respective variable (if prop = TRUE) or NULL (if prop = FALSE),
primes the names of the variables,
type the type of model (1: classification, 2: linear regression, 3: logistic regression),
param further parameters (if addInfo = TRUE in the previous call of logic.bagging),
mat.imp either a matrix containing the improvements due to the variables for each of the models (if addMatImp = TRUE), or NULL (if addMatImp = FALSE),
measure the name of the used importance measure,
useN the value of useN,
threshold NULL if standardize = FALSE, otherwise the $1 - 0.05/m$ quantile of the t-distribution with $B - 1$ degrees of freedom, where $m$ is the number of variables and $B$ is the number of logic regression models composing object,
mu mu (if standardize = TRUE), or NULL (otherwise),
iter iter.

Author(s)

Holger Schwender, <holger.schwender@udo.edu>

References


See Also

logic.bagging, logicFS, vim.logicFS, vim.set, vim.ebam, vim.chisq
Description

Computes the value of the single or the multiple tree measure, respectively, for each prime implicant contained in a logic bagging model to specify the importance of the prime implicant for classification, if the response is binary. If the response is quantitative, the importance is specified by a measure based on the log2-transformed mean square prediction error.

Usage

```r
vim.logicFS(log.out, useN = TRUE, onlyRemove = FALSE, prob.case = 0.5,
            addInfo = FALSE, addMatImp = TRUE)
```

Arguments

- `log.out`: an object of class `logicBagg`, i.e., the output of `logic.bagging`.
- `useN`: logical, specifying if the number of correctly classified out-of-bag observations should be used in the computation of the importance measure. If `FALSE`, the proportion of correctly classified oob observations is used instead.
- `onlyRemove`: should in the single tree case the multiple tree measure be used? If `TRUE`, the prime implicants are only removed from the trees when determining the importance in the single tree case. If `FALSE`, the original single tree measure is computed for each prime implicant, i.e., a prime implicant is not only removed from the trees in which it is contained, but also added to the trees that do not contain this interaction. Ignored in all other than the classification case.
- `prob.case`: a numeric value between 0 and 1. If the logistic regression approach of logic regression is used (i.e., if the response is binary, and in `logic.bagging`, `ntrees` is set to a value larger than 1, or `glm.if.1tree` is set to `TRUE`), then an observation will be classified as a case (or more exactly as 1), if the class probability of this observation estimated by the logic bagging model is larger than `prob.case`.
- `addInfo`: should further information on the logic regression models be added?
- `addMatImp`: should the matrix containing the improvements due to the prime implicants in each of the iterations be added to the output? (For each of the prime implicants, the importance is computed by the average over the `B` improvements.) Must be set to `TRUE`, if standardized importances should be computed using `vim.norm`, or if permutation based importances should be computed using `vim.signperm`.

Value

An object of class `logicFS` containing

- `primes`: the prime implicants,
- `vim`: the importance of the prime implicants,
- `prop`: the proportion of logic regression models containing the prime implicants,
- `type`: the type of model (1: classification, 2: linear regression, 3: logistic regression),
- `param`: further parameters (if `addInfo = TRUE`),
vim.norm

mat.imp the matrix containing the improvements if addMatImp = TRUE, otherwise, NULL,
measure the name of the used importance measure,
useN the value of useN,
threshold NULL,
u

Author(s)
Holger Schwender, <holger.schwender@udo.edu>

References

See Also
logic.bagging, logicFS, vim.norm, vim.signperm

vim.norm | Standardized and Sign-Permutation Based Importance Measure
---------|--------------------------------------------------

Description
Computes a standarized or a sign-permutation based version of either the Single Tree Measure, the Quantitative Response Measure, or the Multiple Tree Measure.

Usage
vim.norm(object, mu = 0)

vim.signperm(object, mu = 0, n.perm = 10000, n.subset = 1000,
version = 1, adjust = "bonferroni", rand = NA)

Arguments
object either the output of logicFS or vim.logicFS with addMatImp = TRUE, or the output of logic.bagging with importance = TRUE and addMatImp = TRUE.
mu a non-negative numeric value against which the importances are tested. See Details.
n.perm the number of sign permutations used in vim.signperm.
n.subset an integer specifying how many permutations should be considered at once.
version either 1 or 2. If 1, then the importance measure is computed by 1 - padj, where padj is the adjusted p-value. If 2, the importance measure is determined by -log10(padj), where a raw p-value equal to 0 is set to 1 / (10 * n.perm) to avoid infinitive importances.
adjust  character vector naming the method with which the raw permutation based p-
values are adjusted for multiplicity. If "qvalue", the function qvalue.cal from
the package siggenes is used to compute q-values. Otherwise, p.adjust is
used to adjust for multiple comparisons. See p.adjust for all other possible
specifications of adjust. If "none", the raw p-values will be used. For more
details, see Details.

rand  an integer for setting the random number generator in a reproducible case.

Details

In both vim.norm and vim.signperm, a paired t-statistic is computed for each prime
implicant, where the numerator is given by $VIM - \mu$ with VIM being the single or the multiple tree
importance, and the denominator is the corresponding standard error computed by employing the $B$

improvements of the considered prime implicant in the $B$ logic regression models, where VIM is the

mean over these $B$ improvements.

Note that in the case of a quantitative response, such a standardization is not necessary. Thus, vim.norm returns a warning when the response is quantitative, and vim.signperm does not divide $VIM - \mu$ by its sample standard error.

Using $\mu = 0$ might lead to calling a prime implicant important, even though it actually shows only
improvements of 1 or 0. When considering the prime implicants, it might be therefore be helpful to
set $\mu$ to a value slightly larger than zero.

In vim.norm, the value of this t-statistic is returned as the standardized importance of a prime
implicant. The larger this value, the more important is the prime implicant. (This applies to all
importance measures - at least for those contained in this package.) Assuming normality, a possible
threshold for a prime implicant to be considered as important is the $1 - 0.05/m$ quantile of the t-
distribution with $B - 1$ degrees of freedom, where $m$ is the number of prime implicants.

In vim.signperm, the sign permutation is used to determine $n.perm$ permuted values of the one-
sample t-statistic, and to compute the raw p-values for each of the prime implicants. Afterwards,
these p-values are adjusted for multiple comparisons using the method specified by adjust. The
permutation based importance of a prime implicant is then given by $1 - \text{these adjusted p-values}.$
Here, a possible threshold for calling a prime implicant important is 0.95.

Value

An object of class logicFS containing

- **primes**: the prime implicants,
- **vim**: the respective importance of the prime implicants,
- **prop**: NULL,
- **type**: the type of model (1: classification, 2: linear regression, 3: logistic regression),
- **param**: further parameters (if addInfo = TRUE),
- **mat.imp**: NULL,
- **measure**: the name of the used importance measure,
- **useN**: the value of useN from the original analysis with, e.g., logicFS,
- **threshold**: the threshold suggested in Details,
- **mu**: $\mu$.

Author(s)

Holger Schwender, <holger.schwender@udo.edu>
References


See Also

logic.bagging, logicFS, vim.logicFS, vim.chisq, vim.ebam

### vim.permSNP

**Permutation Based Importance Measures**

**Description**

Computes the importances of input variables, SNPs, or sets of SNPs, respectively, based on permutations of the response. Currently only available for the classification and the logistic regression approach of logic regression.

**Usage**

```r
vim.permInput(object, n.perm = NULL, standardize = TRUE, rebuild = FALSE, prob.case = 0.5, useAll = FALSE, version = 1, adjust = "bonferroni", addMatPerm = FALSE, rand = NA)
```

```r
vim.permSNP(object, n.perm = NULL, standardize = TRUE, rebuild = FALSE, prob.case = 0.5, useAll = FALSE, version = 1, adjust = "bonferroni", addMatPerm = FALSE, rand = NA)
```

```r
vim.permSet(object, set = NULL, n.perm = NULL, standardize = TRUE, rebuild = FALSE, prob.case = 0.5, useAll = FALSE, version = 1, adjust = "bonferroni", addMatPerm = FALSE, rand = NA)
```

**Arguments**

- `object` an object of class `logicBagg`, i.e. the output of `logic.bagging`.
- `set` either a list or a character or numeric vector.

If `NULL` (default), then it will be assumed that data, i.e. the data set used in the application of `logic.bagging`, has been generated using `make.snp.dummy` or similar functions for coding variables by binary variables, i.e. with a function that splits a variable, say SNPx, into the dummy variables SNPx.1, SNPx.2, ... (where the "." can also be any other sign, e.g., an underscore).

If a character or a numeric vector, then the length of `set` must be equal to the number of variables used in `object`, i.e. the number of columns of data in the `logicBagg` object, and must specify the set to which a variable belongs either by an integer between 1 and the number of sets, or by a set name. If a variable should not be included in any of the sets, set the corresponding entry of `set` to `NA`. Using this specification of `set` it is not possible to assign a variable to more than one sets. For such a case, set `set` to a list (as follows).

If `set` is a list, then each object in this list represents a set of variables. Therefore, each object must be either a character or a numeric vector specifying either the names of the variables that belongs to the respective set or the columns of
data that contains these variables. If names(set) is NULL, generic names will be employed as names for the sets. Otherwise, names(set) are used.

n.perm number of permutations used in the computation of the importances. By default (i.e., if n.perm = NULL), 100 permutations are used if rebuild = TRUE and the regression approach of logic regression has been used in logic.bagging (by setting ntree to an integer larger than 1, or glm.if.1tree = TRUE). Otherwise, 1000 permutation are employed. Note that actually much more permutations should be used.

standardize should the standardized importance measure be used?

rebuild logical indicating whether the logic regression models should be rebuild (i.e., the parameters β of the generalized linear models should be recomputed) after removing a variable or a set of variables from the logic trees and for each permutation of the response. Note that setting rebuild = TRUE increases the computation time substantially.

prob.case a numeric value between 0 and 1. If the logistic regression approach of logic regression has been used in logic.bagging, then an observation will be classified as a case (or more exactly, as 1), if the class probability of this observation is larger than prob.case. Otherwise, prob.case is ignored.

useAll logical indicating whether all m * n.perm permuted values should be used in the computation of the permutation based p-values, where m is the number of variables or sets of variables, respectively. If FALSE, the n.perm permuted values corresponding to the respective variable (or set of variables) are employed in the determination of the p-value of this variable (or set of variables).

version either 1 or 2. If 1, then the importance measure is computed by 1 - padj, where padj is the adjusted p-value. If 2, the importance measure is determined by -log10(padj), where a raw p-value equal to 0 is set to 1 / (10 * n.perm) to avoid infinitive importances.

adjust character vector naming the method with which the raw permutation based p-values are adjusted for multiplicity. If "qvalue", the function qvalue.cal from the package siggenes is used to compute q-values. Otherwise, p.adjust is used to adjust for multiple comparisons. See p.adjust for all other possible specifications of adjust. If "none", the raw p-values will be used.

addMatPerm should the (n.perm + 1) x m matrix containing the original values (first column) and the permuted values (the remaining columns) of the importance measure for the m variables or m sets of variables be added to the output?

rand an integer for setting the random number generator in a reproducible state.

Value

An object of class logicFS containing

vim the values of the importance measure for the input variables, the SNPs, or the sets of SNPs, respectively,

prop NULL,

primes the names of the inputs, SNPs, or sets of variables, respectively,

type the type of model (1: classification, 3: logistic regression),

param NULL,

mat.imp NULL,

measure the name of the used importance measure,
threshold 0.95, i.e., the suggested threshold for calling an input, SNP or set of SNPs, respectively, important (this is just used as default value when plotting the importances, see argument thres of plot.logicFS),
mu NULL,
useN TRUE,
name either "Variable", "SNP", or "Set".
mat.perm if addMatPerm = FALSE, NULL; otherwise, a matrix containing the original and the permuted values of the respective importance measure.

Author(s)
Holger Schwender, <holger.schwender@udo.edu>

References

See Also
logic.bagging, vim.input, vim.set, vim.signperm

vim.set

VIM for SNPs and Sets of Variables

Description
Quantifies the importances of SNPs or sets of variables, respectively, contained in a logic bagging model.

Usage
```r
vim.snp(object, useN = NULL, iter = NULL, standardize = NULL, 
mu = 0, addMatImp = FALSE, prob.case = 0.5, rand = NA)

vim.set(object, set = NULL, useN = NULL, iter = NULL, standardize = NULL, 
mu = 0, addMatImp = FALSE, prob.case = 0.5, rand = NA)
```

Arguments
- `object` an object of class logicBagg, i.e., the output of logic.bagging.
- `set` either a list or a character or numeric vector.
  - If NULL (default), then it will be assumed that data, i.e., the data set used in the application of logic.bagging, has been generated using make.snp.dummy or similar functions for coding variables by binary variables, i.e., with a function that splits a variable, say SNPx, into the dummy variables SNPx.1, SNPx.2, ... (where the "." can also be any other sign, e.g., an underscore).
  - If a character or a numeric vector, then the length of `set` must be equal to the number of variables used in `object`, i.e., the number of columns of data in the logicBagg object, and must specify the set to which a variable belongs either...
by an integer between 1 and the number of sets, or by a set name. If a variable should not be included in any of the sets, set the corresponding entry of set to NA. Using this specification of set it is not possible to assign a variable to more than one sets. For such a case, set set to a list (as follows).

If set is a list, then each object in this list represents a set of variables. Therefore, each object must be either a character or a numeric vector specifying either the names of the variables that belongs to the respective set or the columns of data that contains these variables. If names(set) is NULL, generic names will be employed as names for the sets. Otherwise, names(set) are used.

useN
logical specifying if the number of correctly classified out-of-bag observations should be used in the computation of the importance measure. If FALSE, the proportion of correctly classified oob observations is used instead. If NULL (default), then the specification of useN in object is used.

iter
integer specifying the number of times the values of the variables in the respective set are permuted in the computation of the importance of this set. If NULL (default), the values of the variables are not permuted, but all variables belonging to the set are removed from the model.

standardize
should a standardized version of the importance measure for a set of variables be returned? By default, standardize = TRUE is used in the classification and the (multinomial) logistic regression case, and standardize is set to FALSE in the linear regression case. For details, see mu.

mu
a non-negative numeric value. Ignored if standardize = FALSE. Otherwise, a t-statistic for testing the null hypothesis that the importance of the respective set is equal to mu is computed.

addMatImp
should the matrix containing the improvements due to each of the sets in each of the logic regression models be added to the output?

prob.case
a numeric value between 0 and 1. If the logistic regression approach of logic regression has been used in logic.bagging, then an observation will be classified as a case (or more exactly, as 1), if the class probability of this observation is larger than prob.case. Otherwise, prob.case is ignored.

rand
an integer for setting the random number generator in a reproducible state.

Value
An object of class logicFS containing

vim the importances of the sets of variables,
prop NULL,
primes the names of the sets of variables,
type the type of model (1: classification, 2: linear regression, 3: logistic regression),
param further parameters (if addInfo = TRUE in the previous call of logic.bagging), or NULL (otherwise),
mat.imp either a matrix containing the improvements due to the sets of variables for each of the models (if addMatImp = TRUE), or NULL (if addMatImp = FALSE),
measure the name of the used importance measure,
threshold NULL if standardize = FALSE, otherwise the 1 - 0.95/m quantile of the t-distribution with B - 1 degrees of freedom, where m is the number of sets and B is the number of logic regression models composing object,
mu mu (if standardize = TRUE), or NULL (otherwise),
iter iter.
Author(s)
Holger Schwender, <holger.schwender@udo.edu>

References

See Also
logic.bagging, logicFS, vim.logicFS, vim.input, vim.ebam, vim.chisq
Index

*Topic datasets
  data.logicfs, 2
*Topic hplot
  plot.logicFS, 13
*Topic htest
  logic.oob, 6
  vim.approxPval, 17
  vim.chisq, 18
  vim.input, 21
  vim.logicFS, 23
  vim.norm, 24
  vim.permSNP, 26
  vim.set, 28
*Topic logic
  logic.pimp, 7
  vim.approxPval, 17
  vim.chisq, 18
  vim.ebam, 19
  vim.input, 21
  vim.logicFS, 23
  vim.norm, 24
  vim.permSNP, 26
  vim.set, 28
*Topic manip
  make.snp.dummy, 11
*Topic multivariate
  logicFS, 7
*Topic print
  print.logicFS, 16
*Topic regression
  logic.bagging, 3
  logicFS, 7
  mlogreg, 12
  predict.logicBagg, 15
  predict.mlogreg, 16
*Topic tree
  logic.bagging, 3
  logicFS, 7
  mlogreg, 12
*Topic utilities
  getMatEval, 2
  logic.oob, 6
  logic.pimp, 7
  cl.logicfs (data.logicfs), 2
  data.logics, 2
  getMatEval, 2
  logic.bagging, 2, 3, 6, 7, 10, 11, 13, 15, 18, 19, 21, 22, 24, 26–28, 30
  logic.oob, 6
  logic.pimp, 7
  logicFS, 2, 5, 7, 7, 11, 13, 15, 17–19, 21, 22, 24–26, 30
  make.snp.dummy, 4, 9, 11, 12, 26, 28
  mlogreg, 3, 4, 8, 9, 12, 16
  plot.logicBagg, 5
  plot.logicBagg (plot.logicFS), 13
  plot.logicFS, 10, 13, 28
  predict.logicBagg, 5, 15
  predict.mlogreg, 13, 16
  prime.implicants, 7
  print.logicBagg (logic.bagging), 3
  print.logicFS, 16
  print.mlogreg (mlogreg), 12
  vim.approxPval, 17
  vim.chisq, 18, 21, 22, 26, 30
  vim.ebam, 19, 19, 22, 26, 30
  vim.individual (vim.input), 21
  vim.input, 18, 21, 28, 30
  vim.logicFS, 17, 19–22, 23, 24, 26, 30
  vim.norm, 4, 9, 19–21, 23, 24, 24
  vim.permInput (vim.permSNP), 26
  vim.permSet, 18
  vim.permSet (vim.permSNP), 26
  vim.permSNP, 26
  vim.set, 18, 22, 28, 28
  vim.signperm, 4, 9, 19, 20, 23, 24, 28
  vim.signperm (vim.norm), 24
  vim.snp (vim.set), 28