Package 'MXM'

August 5, 2014

Type Package

Title Discovering multiple, statistically-equivalent signatures

Version 0.2.1

URL http://www.mensxmachina.org/

Date 2014-05-6

Author Ioannis Tsamardinos, Vincenzo Lagani, Giorgos Borboudakis, Giorgos Athineou

Maintainer Giorgos Athineou <athineou@ics.forth.gr>

Description Feature selection methods for identifying minimal, statistically-equivalent and equally-predictive feature subsets. MXM stands for ``Mens eX Machina", meaning ``Mind from the Machine" in Latin.

License GPL-2

Suggests gRbase, hash, Biobase, VGAM, MASS, pcalg, survival, methods

NeedsCompilation yes

Repository CRAN

Date/Publication 2014-08-05 00:35:02

R topics documented:

MXM-package	*This is an R package that implements feature selection methods for identifying minimal, statistically-equivalent and equally-predictive feature subsets.*

Description

The 'MXM' (Mens eX Machina, meaning 'Mind from the Machine' in Latin) package provides source code for the SES algorithm and for some appropriate statistical conditional independence tests (testIndFisher, testIndLogistic, gSquare and censIndLR are included). Read the package's help pages for more details.

Details

Package:	MXM
Type:	Package
Version:	0.2.1
Date:	2014-8-04
License:	GPL-2

Maintainers

Giorgos Athineou <athineou@ics.forth.gr>, Vincenzo Lagani <vlagani@ics.forth.gr>

Author(s)

Giorgos Athineou <athineou@ics.forth.gr>, Vincenzo Lagani <vlagani@ics.forth.gr>, Giorgos Borboudakis <borbudak@ics.forth.gr>, Ioannis Tsamardinos <tsamard@ics.forth.gr>

References

I. Tsamardinos, V. Lagani and D. Pappas (2012) Discovering multiple, equivalent biomarker signatures. In proceedings of the 7th conference of the Hellenic Society for Computational Biology & Bioinformatics - HSCBB12.

See Also

SES, censIndLR, testIndFisher, testIndLogistic, gSquare, censIndLR

censIndLR *Conditional independence test based on the Log Likelihood ratio test for survival data (see reference below)*

Description

The main task of this test is to provide a p-value PVALUE for the null hypothesis: feature 'X' is independent from 'TARGET' given a conditioning set CS. This test is based on the widely used Cox regression model (Cox, 1972).

Usage

```
censIndLR(target, dataset, xIndex, csIndex, dataInfo = NULL, univariateModels = NULL,
hash = FALSE, stat_hash = NULL, pvalue_hash = NULL)
```

Arguments

target	A Survival object (class Surv from package survival) containing the time to event data (time) and the status indicator vector (event). View Surv documentation for more information.
dataset	A numeric data matrix containing the variables for performing the test. Rows as samples and columns as features.
xIndex	The index of the variable whose association with the target we want to test.
csIndex	The indices of the variables to condition on.
dataInfo	list object with information on the structure of the data. Default value is NULL.
univariateModels	Fast alternative to the hash object for univariate test. List with vectors "pvalues" (p-values), "stats" (statistics) and "flags" (flag = TRUE if the test was succesful) representing the univariate association of each variable with the target. Default value is NULL.
hash	A boolean variable which indicates whether (TRUE) or not (FALSE) to use the hash-based implementation of the statistics of SES. Default value is FALSE. If TRUE you have to specify the stat_hash argument and the pvalue_hash argument.
stat_hash	A hash object (hash package required) which contains the cached generated statistics of a SES run in the current dataset, using the current test.
pvalue_hash	A hash object (hash package required) which contains the cached generated p-values of a SES run in the current dataset, using the current test.

Details

If hash = TRUE, testIndLogistic requires the arguments 'stat_hash' and 'pvalue_hash' for the hash-based implementation of the statistic test. These hash Objects are produced or updated by each run of SES (if hash == TRUE) and they can be reused in order to speed up next runs of the current

statistic test. If "SESoutput" is the output of a SES run, then these objects can be retrieved by SESoutput@hashObject$stat_hash and the SESoutput@hashObject$pvalue_hash.

Important: Use these arguments only with the same dataset that was used at initialization.

Value

A list including:

pvalue	A numeric value that represents the generated p-value.
stat	A numeric value that represents the generated statistic.
flag	A numeric value (control flag) which indicates whether the test was succesful (0) or not (1).
stat_hash	The current hash object used for the statistics. See argument stat_hash and details. If argument hash = FALSE this is NULL.
pvalue_hash	The current hash object used for the p-values. See argument stat_hash and details. If argument hash = FALSE this is NULL.

Note

This test uses the functions coxph and Surv of the package survival and the function anova (analysis of variance) of the package stats.

Author(s)

R implementation and documentation: Vincenzo Lagani <vlagani@ics.forth.gr>, Giorgos Athineou <athineou@ics.forth.gr>

References

V. Lagani and I. Tsamardinos (2010). Structure-based variable selection for survival data. Bioinformatics Journal 16(15): 1887-1894.

Cox,D.R. (1972) Regression models and life-tables. J. R. Stat. Soc., 34, 187-220.

See Also

SES, testIndFisher, gSquare, testIndLogistic, Surv, coxph, anova

Examples

```
#create a survival simulated dataset
dataset <- matrix(nrow = 1000 , ncol = 100)
dataset <- apply(dataset, 1:2, function(i) runif(1, 1, 100))
dataset <- as.data.frame(dataset);
timeToEvent = rep(0,1000)
event = rep(0,1000)
c = rep(0,1000)
for(i in 1:1000)
{
  timeToEvent[i] = dataset[i,1] + 0.5*dataset[i,30] + 2*dataset[i,65] + runif(1, 0, 1);
```

```
  event[i] = sample(c(0,1),1)
  c[i] = runif(1, 0, timeToEvent[i]-0.5)
  if(event[i] == 0)
  {
    timeToEvent[i] = timeToEvent[i] - c[i]
  }
}

#init the Surv object class feature
require(survival)
target <- Surv(time=timeToEvent, event=event)

#run the censIndLR   conditional independence test
require(stats)
res = censIndLR(target, dataset, xIndex=12, csIndex=c(35,7,4))
res

#run the SES algorithm using the censIndLR conditional independence
#test for the survival class variable

#require(gRbase) #for faster computations in the internal functions
sesObject <- SES(target , dataset , max_k=1 , threshold=0.05 , test="censIndLR");
#print summary of the SES output
summary(sesObject);
#plot the SES output
plot(sesObject, mode="all");
```

`gSquare` *G square conditional independence test for discrete data based on the log likelihood ratio test.*

Description

The main task of this test is to provide a p-value PVALUE for the null hypothesis: feature 'X' is independent from 'TARGET' given a conditioning set CS. This test is based on the log likelihood ratio test.

Usage

```
gSquare(target, dataset, xIndex, csIndex, dataInfo = NULL, univariateModels = NULL,
hash = FALSE, stat_hash = NULL, pvalue_hash = NULL)
```

Arguments

`target`	A numeric vector containing the values of the target variable.
`dataset`	A numeric data matrix containing the variables for performing the test. Rows as samples and columns as features.
`xIndex`	The index of the variable whose association with the target we want to test.

`csIndex` The indices of the variables to condition on.

`dataInfo` list object with information on the structure of the data. Default value is NULL.

`univariateModels`
Fast alternative to the hash object for univariate test. List with vectors "pvalues" (p-values), "stats" (statistics) and "flags" (flag = TRUE if the test was succesful) representing the univariate association of each variable with the target. Default value is NULL.

`hash` A boolean variable which indicates whether (TRUE) or not (FALSE) to use the hash-based implementation of the statistics of SES. Default value is FALSE. If TRUE you have to specify the stat_hash argument and the pvalue_hash argument.

`stat_hash` A hash object (hash package required) which contains the cached generated statistics of a SES run in the current dataset, using the current test.

`pvalue_hash` A hash object (hash package required) which contains the cached generated p-values of a SES run in the current dataset, using the current test.

Details

If hash = TRUE, testIndLogistic requires the arguments 'stat_hash' and 'pvalue_hash' for the hash-based implementation of the statistic test. These hash Objects are produced or updated by each run of SES (if hash == TRUE) and they can be reused in order to speed up next runs of the current statistic test. If "SESoutput" is the output of a SES run, then these objects can be retrieved by SESoutput@hashObject$stat_hash and the SESoutput@hashObject$pvalue_hash.

Important: Use these arguments only with the same dataset that was used at initialization.

Value

A list including:

`pvalue` A numeric value that represents the generated p-value due to Fisher's method (see reference below).

`stat` A numeric value that represents the generated statistic due to Fisher's method (see reference below).

`flag` A numeric value (control flag) which indicates whether the test was succesful (0) or not (1).

`stat_hash` The current hash object used for the statistics. See argument stat_hash and details. If argument hash = FALSE this is NULL.

`pvalue_hash` The current hash object used for the p-values. See argument stat_hash and details. If argument hash = FALSE this is NULL.

Note

This test uses the functions gSquareBin and gSquareDis of the package pcalg in order to generate the pvalue for the G^2 test.

Author(s)

R implementation and documentation: Giorgos Athineou <athineou@ics.forth.gr>

See Also

SES, testIndFisher, testIndLogistic, censIndLR

Examples

```
#simulate a dataset with binary data
dataset <- matrix(nrow = 50 , ncol = 101)
dataset <- apply(dataset, 2, function(i) sample(c(0,1),50, replace=TRUE))
#initialize binary target
target <- dataset[,101]
#remove target from the dataset
dataset <- dataset[,-101]
#run the gSquare conditional independence test for the binary class variable
require(pcalg)
results <- gSquare(target, dataset, xIndex = 44, csIndex = c(10,20))
results

#require(gRbase) #for faster computations in the internal functions
#run the SES algorithm using the gSquare conditional independence test for the binary class variable
sesObject <- SES(target , dataset , max_k=3 , threshold=0.05 , test="gSquare");
#print summary of the SES output
summary(sesObject);
#plot the SES output
plot(sesObject, mode="all");
```

MXM-internal *Internal MXM Functions*

Description

Internal functions of Package **MXM**

Details

These functions are only for internal usage of the MXM package - NOT to be called by the user.

For faster computations in the internal SES functions, install the suggested package "**gRbase**".

Functions

- InternalSES(...)
- IdentifyEquivalence(...)
- apply_ideq(...)
- compare_p_values(...)

- identifyTheEquivalent(...)
- max_min_assoc(...)
- min_assoc(...)
- nchoosekm(...)
- univariateScore(...)

SES *Feature selection algorithm for identifying multiple minimal, statistically-equivalent and equally-predictive feature signatures.*

Description

SES algorithm follows a forward-backward filter approach for feature selection in order to provide minimal, highly-predictive, statistically-equivalent, multiple feature subsets of a high dimensional dataset. See also Details.

Usage

```
SES(target = NULL, dataset = NULL, max_k = 3, threshold = 0.05, test = NULL,
 user_test = NULL, hash = FALSE, hashObject = NULL)
```

Arguments

target	The class variable. Provide either a string, an integer value, a vector, a factor, an ordered factor or a Surv object. See also Details.
dataset	The data-set; provide either a data frame or a matrix (columns = variables , rows = samples). Alternatively, provide an ExpressionSet (in which case rows are samples and columns are features, see bioconductor for details).
max_k	The maximum conditioning set to use in the conditional indepedence test (see Details). Integer, default value is 3.
threshold	Threshold (suitable values in [0,1]) for assessing p-values significance. Default value is 0.05.
test	The conditional independence test to use. Default value is NULL. Available conditional independence tests: • "testIndFisher": Fisher conditional independence test for continous targets. • "testIndLogistic": Conditional Independence Test based on logistic regression for binary, categorical and ordinal targets. • "gSquare": Conditional Independence test based on the G test of independence (log likelihood ratio test). • "censIndLR": Conditional independence test for survival data based on the Log likelihood ratio test. See also Details.

`user_test` A user-defined conditional independence test (provide a closure type object). Default value is NULL. If this is defined, the "test" argument is ignored.

`hash` A boolean variable which indicates whether (TRUE) or not (FALSE) to store the statistics calculated during SES execution in a hash-type object. Default value is FALSE. If TRUE a hashObject is produced.

`hashObject` A List with the hash objects generated in a previous run of SES. Each time SES runs with "hash=TRUE" it produces a list of hashObjects that can be re-used in order to speed up next runs of SES.

Important: the generated hashObjects should be used only when the same dataset is re-analyzed, possibly with different values of max_k and threshold.

Details

This function implements the Statistically Equivalent Signature (SES) algorithm as presented in "Tsamardinos, Lagani and Pappas, HSCBB 2012"

(http://www.mensxmachina.org/publications/discovering-multiple-equivalent-biomarker-signatures/)

For faster computations in the internal SES functions, install the suggested package "**gRbase**".

The max_k option: the maximum size of the conditioning set to use in the conditioning independence test. Larger values provide more accurate results, at the cost of higher computational times. When the sample size is small (e.g., <50 samples) the max_k parameter should be <=5, otherwise the conditional independence test may not be able to provide reliable results.

If the dataset contains missing (NA) values, they will automatically be replaced by the current variable (column) mean value with an appropriate warning to the user after the execution.

If the target is a single integer value or a string, it has to corresponds to the column number or to the name of the target feature in the dataset. In any other case the target is a variable that is not contained in the dataset.

If the current 'test' argument is defined as NULL or "auto" and the user_test argument is NULL then the algorithm automatically selects the best test based on the type of the data. Particularly:

- if target is a factor, the multinomial logistic test is used
- if target is a ordered factor, the ordered logit regression is used in the logistic test
- if target is a numerical vector, the fisher conditional independence test is used
- if target is a Surv object, the Survival conditional independence test is used

Conditional independence test functions to be pass through the user_test argument should have the same signature of the included test. See "?testIndFisher" for an example.

Value

The output of the algorithm is an object of the class 'SESoutput' including:

`selectedVars` The selected variables, i.e., the signature of the target variable.

`selectedVarsOrder`
The order of the selected variables according to increasing pvalues.

`queues` A list containing a list (queue) of equivalent features for each variable included in selectedVars. An equivalent signature can be built by selecting a single feature from each queue.

`signatures` A matrix reporting all equivalent signatures (one signature for each row).

`hashObject` The hashObject caching the statistic calculted in the current run.

`pvalues` For each feature included in the dataset, this vector reports the strength of its association with the target in the context of all other variables. Particularly, this vector reports the max p-values foudn when the association of each variable with the target is tested against different conditional sets. Lower values indicate higher association.

`stats` The statistics corresponding to "pvalues" (higher values indicates higher association).

`max_k` The max_k option used in the current run.

`threshold` The threshold option used in the current run.

`runtime` The run time of the algorithm. A numeric vector. The first element is the user time, the seond element is the system time and the third element is the elapsed time.

Generic Functions implemented for SESoutput Object:

`summary(x=SESoutput)`
: Summary view of the SESoutput object.

`plot(object=SESoutput, mode="all")`
: Plots the generated pvalues (using barplot) of the current SESoutput object in comparison to the threshold.

Argument mode can be either "all" or "partial" for the first 500 pvalues of the object.

Note

The packages required for the SES algorithm operations are:

gRbase : for faster computations in the internal functions

hash : for the hash-based implementation

VGAM : require(stats) and require(MASS) for the testIndLogistic test

survival : for the censIndLR test

pcalg : for the gSquare test.

Author(s)

Ioannis Tsamardinos, Vincenzo Lagani (Copyright 2013)

R implementation and documentation: Giorgos Athineou <athineou@ics.forth.gr> Vincenzo Lagani <vlagani@ics.forth.gr>

References

I. Tsamardinos, V. Lagani and D. Pappas (2012) Discovering multiple, equivalent biomarker signatures. In proceedings of the 7th conference of the Hellenic Society for Computational Biology & Bioinformatics - HSCBB12.

See Also

testIndFisher, testIndLogistic, gSquare, censIndLR

Examples

```
set.seed(123)
#require(gRbase) #for faster computations in the internal functions
require(hash)

#simulate a dataset with continuous data
dataset <- matrix(nrow = 1000 , ncol = 300)
dataset <- apply(dataset, 1:2, function(i) runif(1, 1, 100))

#define a simulated class variable
target = 3*dataset[,10] + 2*dataset[,200] + 3*dataset[,20] + runif(1, 0, 1);

#define some simulated equivalences
dataset[,15] = dataset[,10]
dataset[,250] = dataset[,200]
dataset[,230] = dataset[,200]

#run the SES algorithm
sesObject <- SES(target , dataset , max_k=5 , threshold=0.2 , test="testIndFisher",
hash = TRUE, hashObject=NULL);
#print summary of the SES output
summary(sesObject);
#plot the SES output
plot(sesObject, mode="all");
#get the queues with the equivalences for each selected variable
sesObject@queues
#get the generated signatures
sesObject@signatures;
#get the run time
# > sesObject@runtime;
# user  system elapsed
# 0.35    0.00    0.35


#re-run the SES algorithm with the same or different configuration
#under the hash-based implementation of retrieving the statistics
#in the SAME dataset (!important)
hashObj <- sesObject@hashObject;
sesObject2 <- SES(target , dataset , max_k=2 , threshold=0.01 , test="testIndFisher",
hash = TRUE, hashObject=hashObj);
#retrieve the results: summary, plot, sesObject2@...)
summary(sesObject2)
#get the run time
# > sesObject2@runtime;
# user  system elapsed
# 0.01    0.00    0.01
```

SESoutput-class *Class* "SESoutput"

Description

SES output object class.

Objects from the Class

Objects can be created by calls of the form `new("SESoutput", ...)`.

Slots

`selectedVars`: Object of class "numeric"

`selectedVarsOrder`: Object of class "numeric"

`queues`: Object of class "list"

`signatures`: Object of class "matrix"

`hashObject`: Object of class "list"

`pvalues`: Object of class "numeric"

`stats`: Object of class "numeric"

`max_k`: Object of class "numeric"

`threshold`: Object of class "numeric"

`runtime`: Object of class "proc_time"

Methods

summary `summary(object = "SESoutput")`: Generic function for summarizing the results of the SES output

plot `plot(x = "SESoutput", mode = "all")`: Generic function for plotting the generated pvalues of the SESoutput object. Argument mode = "all" for plotting all the pvalues or mode="partial" for partial plotting the first 500 pvalues

Author(s)

Giorgos Athineou <athineou@ics.forth.gr>

See Also

SES

Examples

```
showClass("SESoutput")
```

testIndFisher *Fisher's conditional independence test for continous class variables.*

Description

The main task of this test is to provide a p-value PVALUE for the null hypothesis: feature 'X' is independent from 'TARGET' given a conditioning set CS

Usage

```
testIndFisher(target, dataset, xIndex, csIndex, dataInfo = NULL, univariateModels = NULL,
hash = FALSE, stat_hash = NULL, pvalue_hash = NULL)
```

Arguments

target	A numeric vector containing the values of the target variable.
dataset	A numeric data matrix containing the variables for performing the test. Rows as samples and columns as features.
xIndex	The index of the variable whose association with the target we want to test.
csIndex	The indices of the variables to condition on.
dataInfo	list object with information on the structure of the data. Default value is NULL.
univariateModels	Fast alternative to the hash object for univariate test. List with vectors "pvalues" (p-values), "stats" (statistics) and "flags" (flag = TRUE if the test was succesful) representing the univariate association of each variable with the target. Default value is NULL.
hash	A boolean variable which indicates whether (TRUE) or not (FALSE) to use tha hash-based implementation of the statistics of SES. Default value is FALSE. If TRUE you have to specify the stat_hash argument and the pvalue_hash argument.
stat_hash	A hash object (hash package required) which contains the cached generated statistics of a SES run in the current dataset, using the current test.
pvalue_hash	A hash object (hash package required) which contains the cached generated p-values of a SES run in the current dataset, using the current test.

Details

If hash = TRUE, testIndLogistic requires the arguments 'stat_hash' and 'pvalue_hash' for the hash-based implementation of the statistic test. These hash Objects are produced or updated by each run of SES (if hash == TRUE) and they can be reused in order to speed up next runs of the current statistic test. If "SESoutput" is the output of a SES run, then these objects can be retrieved by SESoutput@hashObject$stat_hash and the SESoutput@hashObject$pvalue_hash.

Important: Use these arguments only with the same dataset that was used at initialization.

Value

A list including:

pvalue	A numeric value that represents the generated p-value due to Fisher's method (see reference below).
stat	A numeric value that represents the generated statistic due to Fisher's method (see reference below).
flag	A numeric value (control flag) which indicates whether the test was succesful (0) or not (1).
stat_hash	The current hash object used for the statistics. See argument stat_hash and details. If argument hash = FALSE this is NULL.
pvalue_hash	The current hash object used for the p-values. See argument stat_hash and details. If argument hash = FALSE this is NULL.

Author(s)

Vincenzo Lagani and Ioannis Tsamardinos (Copyright 2012)

R implementation and documentation: Giorgos Athineou <athineou@ics.forth.gr> Vincenzo Lagani <vlagani@ics.forth.gr>

References

Peter Spirtes, Clark Glymour, and Richard Scheines. Causation, Prediction, and Search. The MIT Press, Cambridge, MA, USA, second edition, January 2001.

See Also

SES, testIndLogistic, gSquare, censIndLR

Examples

```
#simulate a dataset with continuous data
dataset <- matrix(nrow = 1000 , ncol = 200)
dataset <- apply(dataset, 1:2, function(i) runif(1, 1, 100))
#the target feature is the last column of the dataset as a vector
target <- dataset[,200]
results <- testIndFisher(target, dataset, xIndex = 44, csIndex = 100)
#>results
#   $pvalue
#   [1] 0.5553586
#
#   $stat
#   [1] 0.01869107
#
#   $flag
#   [1] 1
#
#   $stat_hash
```

```
#   NULL
#
#   $pvalue_hash
#   NULL

#require(gRbase)  #for faster computations in the internal functions
#define class variable (here tha last column of the dataset)
target = 200;
#run the SES algorithm using the testIndFisher conditional independence test
sesObject <- SES(target , dataset , max_k=3 , threshold=0.05 , test="testIndFisher");
#print summary of the SES output
summary(sesObject);
#plot the SES output
plot(sesObject, mode="all");
```

`testIndLogistic` *Conditional independence test based on logistic regression for binary, categorical or ordinal class variables.*

Description

The main task of this test is to provide a p-value PVALUE for the null hypothesis: feature 'X' is independent from 'TARGET' given a conditioning set CS. The pvalue is calculated by comparing a logistic model based on the conditioning set CS against a model whose regressor are both X and CS. The comparison is performed through a chi-square test with one degree of freedom on the difference between the deviances of the two models.

Usage

```
testIndLogistic(target,dataset,xIndex,csIndex,dataInfo = NULL,univariateModels = NULL,
hash = FALSE, stat_hash = NULL, pvalue_hash = NULL, target_type = 0)
```

Arguments

`target`	A numeric vector containing the values of the target variable.
`dataset`	A numeric data matrix containing the variables for performing the test. Rows as samples and columns as features.
`xIndex`	The index of the variable whose association with the target we want to test.
`csIndex`	The indices of the variables to condition on.
`dataInfo`	list object with information on the structure of the data. Default value is NULL.
`univariateModels`	Fast alternative to the hash object for univariate test. List with vectors "pvalues" (p-values), "stats" (statistics) and "flags" (flag = TRUE if the test was succesful) representing the univariate association of each variable with the target. Default value is NULL.

hash	A boolean variable which indicates whether (TRUE) or not (FALSE) to use the hash-based implementation of the statistics of SES. Default value is FALSE. If TRUE you have to specify the stat_hash argument and the pvalue_hash argument.
stat_hash	A hash object (hash package required) which contains the cached generated statistics of a SES run in the current dataset, using the current test.
pvalue_hash	A hash object (hash package required) which contains the cached generated p-values of a SES run in the current dataset, using the current test.
target_type	A numeric vector that represents the type of the target. Default value is 0. See details for more. • target_type = 1 (binary target) • target_type = 2 (nominal target) • target_type = 3 (ordinal target)

Details

If argument target_type=0 then testIndLogistic requires the dataInfo argument to indicate the type of the current target:

- dataInfo$target_type = "binary" (binary target)
- dataInfo$target_type = "nominal" (nominal target)
- dataInfo$target_type = "ordinal" (ordinal target)

If hash = TRUE, testIndLogistic requires the arguments 'stat_hash' and 'pvalue_hash' for the hash-based implementation of the statistic test. These hash Objects are produced or updated by each run of SES (if hash == TRUE) and they can be reused in order to speed up next runs of the current statistic test. If "SESoutput" is the output of a SES run, then these objects can be retrieved by SESoutput@hashObject$stat_hash and the SESoutput@hashObject$pvalue_hash.

Important: Use these arguments only with the same dataset that was used at initialization.

Value

A list including:

pvalue	A numeric value that represents the generated p-value.
stat	A numeric value that represents the generated statistic.
flag	A numeric value (control flag) which indicates whether the test was succesful (0) or not (1).
stat_hash	The current hash object used for the statistics. See argument stat_hash and details. If argument hash = FALSE this is NULL.
pvalue_hash	The current hash object used for the p-values. See argument stat_hash and details. If argument hash = FALSE this is NULL.

Note

This test uses the function vglm (package VGAM) for multinomial logistic regression, the function polr (package MASS) for ordinal logit regression and the function glm (package stats) for binomial regression.

Author(s)

Vincenzo Lagani and Ioannis Tsamardinos (Copyright 2012)

R implementation and documentation: Vincenzo Lagani <vlagani@ics.forth.gr> Giorgos Athineou <athineou@ics.forth.gr>

References

Vincenzo Lagani and Ioannis Tsamardinos (2010), Structure-based Variable Selection for Survival Data, Bioinformatics 26(15):1887-1894.

See Also

SES, testIndFisher, gSquare, censIndLR

Examples

```
#require(gRbase) #for faster computations in the internal functions
require(VGAM)
require(MASS)
require(stats)

#simulate a dataset with categorical data
dataset_m <- matrix(nrow = 20 , ncol = 51)
dataset_m <- apply(dataset_m, 2, function(i) sample(c(0,1,2),50, replace=TRUE))
#initialize categorical target
target_m <- dataset_m[,51]
#remove target from the dataset
dataset_m <- dataset_m[,-51]
#run the conditional independence test for the nominal class variable
results_m <- testIndLogistic(target_m, dataset_m, xIndex = 44, csIndex = c(10,20), target_type=2)
results_m

require(VGAM)
#run the SES algorithm using the testIndLogistic conditional independence test
#for the nominal class variable
sesObject <- SES(as.factor(target_m), dataset_m, max_k=3 ,threshold=0.05 ,test="testIndLogistic");
#print summary of the SES output
summary(sesObject);
#plot the SES output
plot(sesObject, mode="all");

########################################################################

#run the conditional independence test for the ordinal class variable
results_o <- testIndLogistic(target_m, dataset_m, xIndex = 44, csIndex = c(10,20), target_type=3)
results_o

require(MASS)
#run the SES algorithm using the testIndLogistic conditional independence test
#for the ordinal class variable
sesObject <- SES(factor(target_m, ordered=TRUE) , dataset_m , max_k=3 , threshold=0.05 ,
```

```
                test="testIndLogistic");
#print summary of the SES output
summary(sesObject);
#plot the SES output
plot(sesObject, mode="all");

#############################################################################

#simulate a dataset with binary data
dataset_b <- matrix(nrow = 20 , ncol = 51)
dataset_b <- apply(dataset_b, 2, function(i) sample(c(0,1),50, replace=TRUE))
#initialize binary target
target_b <- dataset_b[,51]
#remove target from the dataset
dataset_b <- dataset_b[,-51]
#run the conditional independence test for the binary class variable
results_b <- testIndLogistic(target_b, dataset_b, xIndex = 44, csIndex = c(10,20), target_type=1)
results_b

require(stats)
#run the SES algorithm using the testIndLogistic conditional independence test
#for the binary class variable
sesObject <- SES(factor(target_b) , dataset_m , max_k=3 , threshold=0.05 , test="testIndLogistic");
#print summary of the SES output
summary(sesObject);
#plot the SES output
plot(sesObject, mode="all");
```

Index