Package ‘MXM’

February 20, 2017

Type Package

Title Discovering Multiple, Statistically-Equivalent Signatures

Version 0.9.8

URL http://mensxmachina.org

Date 2017-02-07

Author Ioannis Tsamardinos, Vincenzo Lagani, Giorgos Athineou, Michail Tsagris, Giorgos Borboudakis, Anna Roumpelaki

Maintainer Michail Tsagris <mtsagris@csd.uoc.gr>

Description Feature selection methods for identifying minimal, statistically-equivalent and equally-predictive feature subsets. Bayesian network algorithms and related functions are also included. The package name 'MXM' stands for ``Mens eX Machina'', meaning ``Mind from the Machine'' in Latin.

License GPL-2

Suggests hash, R.rsp,

VignetteBuilder R.rsp,

Imports methods, stats, utils, survival, MASS, graphics, ordinal, nnet, quantreg, lme4, foreach, doParallel, parallel, speedglm, e1071, relations, Rfast, visNetwork, energy

NeedsCompilation no

Repository CRAN

Date/Publication 2017-02-20 15:12:09

R topics documented:

`MXM-package` *This is an R package that currently implements feature selection methods for identifying minimal, statistically-equivalent and equally-predictive feature subsets. In addition, two algorithms for constructing the skeleton of a Bayesian network are included.*

Description

'MXM' stands for Mens eX Machina, meaning 'Mind from the Machine' in Latin. The package provides source code for the SES algorithm and for some appropriate statistical conditional independence tests. (Fisher and Spearman corelation, G-square test are some examples. Currently the response variable can be univariate or multivariate Euclidean, proportions within 0 and 1, compositional data without zeros and ones, binary, nominal or ordinal multinomial, count data (handling also overdispersed and with more zeros than expected), longitudinal, clustered data, survival and case-control. Robust versions are also available in some cases and a K-fold cross validation is offered. Bayesian network related algorithms and ridge reression are also included. Read the package's help pages for more details.

MMPC and SES can handle even thousands of variables and for some tests, even many sample sizes of tens of thousands. The user is best advised to check his variables in the beginning. For some regressions, logistic and Poisson for example, we have used C++ codes for speed reasons. Thus no check is done for a variable with zero variance for instance. Something like colVars could be used in the first place to remove variables with zero variance.

Details

Package:	MXM
Type:	Package
Version:	0.9.8
Date:	2017-02-07
License:	GPL-2

Maintainer

Michail Tsagris <mtsagris@csd.uoc.gr>

Note

Acknowledgments: European Research Council (ERC) project No 617393, "CAUSALPATH - Next Generation Causal Analysis" for funding.

Michail Tsagris would like to express his acknowledgments to Marios Dimitriadis and Manos Papadakis, undergraduate students in the department of computer science, university of Crete, for their programming tips and advice.

Author(s)

Giorgos Athineou <athineou@csd.uoc.gr>, Vincenzo Lagani <vlagani@csd.uoc.gr>, Michail Tsagris <mtsagris@csd.uoc.gr>, Giorgos Borboudakis <borbudak@csd.uoc.gr>, Ioannis Tsamardinos <tsamard@csd.uoc.gr>, Anna Roumpelaki <anna.roumpelaki@gmail.com>

References

Tsamardinos, Brown and Aliferis (2006). The max-min hill-climbing Bayesian network structure learning algorithm. Machine learning, 65(1), 31-78.

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.

Tsamardinos I., Aliferis C. F. and Statnikov, A. (2003). Time and sample efficient discovery of Markov blankets and direct causal relations. In Proceedings of the 9th ACM SIGKDD international conference on Knowledge discovery and data mining p. 673-678.

See Also

`SES`, `MMPC`, `censIndCR`,`testIndFisher`, `testIndLogistic`, `gSquare`, `testIndRQ`

Ancestors and descendants of a node in a directed graph

Returns and plots, if asked, the descendants or ancestors of one or all node(s) (or variable(s))

Description

Returns and plots, if asked, the descendants or ancestors of one or all node(s) (or variable(s))

Usage

```
findDescendants(G, node = NULL, graph = FALSE)
findAncestors(G, node = NULL, graph = FALSE)
```

Arguments

`G`	The graph matrix as produced from `pc.or` or any other algorithm which produces directed graphs.
`node`	A numerical value indicating the node (or variable) whose descendants are to be returned.
`graph`	A boolean variable. If TRUE the relevant graph will appear (if there are descendants).

Details

The functions searches for the descendants of some node. This is an S3 class output.

Value

`isAnc`	A matrix of the same dimensions as the original graph matrix with 0s and 1s. isAnc[i, j] = 1 indicates that the i-th node is an ancestor of the j-th node. If the argument "node" is NULL, only this matrix will be returned.
`Ganc`	A matrix of dimensions equal to the number of descendants of the node with 0s and 1s, if the argument "node" is not NULL.
`anc`	The descendants of the node if the argument "node" is not NULL.

Author(s)

Anna Roumpelaki

R implementation and documentation: Anna Roumpelaki <anna.roumpelaki@gmail.com>

See Also

`plotnetwork`, `nei`, `mb`, `pc.or`

Examples

```
# simulate a dataset with continuous data
# simulate a dataset with continuous data
y = rdag(1000, 10, 0.3)
tru = y$G
x = y$x
mod = pc.con(x)
G = pc.or(mod)$G
plotnetwork(G)
findDescendants(G, 4, graph = FALSE)
findAncestors(G, 4, graph = FALSE)
findAncestors(G)
```

Backward selection regression

Variable selection in regression models with backward selection

Description

Variable selection in regression models with backward selection

Usage

```
bs.reg(target, dataset, threshold = 0.05, wei = NULL, test = NULL, user_test = NULL,
robust = FALSE)
```

Arguments

`target`	The class variable. Provide either a string, an integer, a numeric value, a vector, a factor, an ordered factor or a Surv object. See also Details.
`dataset`	The dataset; provide either a data frame or a matrix (columns = variables, rows = samples). In either case, only two cases are avaialble, either all data are continuous, or categorical.
`threshold`	Threshold (suitable values in [0,1]) for assessing p-values significance. Default value is 0.05.
`test`	The regression model to use. Available options are most of the tests for SES and MMPC. The ones NOT available are "gSquare", "censIndER", "testIndMVreg", "testIndClogit", "testIndSpearman" and "testIndFisher". If you want to use multinomial or ordinal logistic regression, make sure your target is factor. See also `SES` and `CondIndTests` for the tests.
`wei`	A vector of weights to be used for weighted regression. The default value is NULL. It is not suggested when robust is set to TRUE.
`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.
`robust`	A boolean variable which indicates whether (TRUE) or not (FALSE) to use a robust version of the statistical test if it is available. It takes more time than a non robust version but it is suggested in case of outliers. Default value is FALSE.

Details

This functions currently implements only linear, binary logistic and Poisson regression. If the sample size is less than the number of variables a meesage will appear and no backward regression is performed.

Value

The output of the algorithm is S3 object including:

`runtime`	The run time of the algorithm. A numeric vector. The first element is the user time, the second element is the system time and the third element is the elapsed time.
`info`	A matrix with the non selected variables and their latest test statistics and p-values.
`mat`	A matrix with the selected variables and their latest statistics and p-values.
`ci_test`	The conditional independence test used.
`final`	The final regression model.

Author(s)

Michail Tsagris

R implementation and documentation: Giorgos Athineou <athineou@csd.uoc.gr> Michail Tsagris <mtsagris@csd.uoc.gr>

See Also

`glm.fsreg`, `lm.fsreg`, `bic.fsreg`, `bic.glm.fsreg`, `CondIndTests`, `MMPC`, `SES`

Examples

```
set.seed(123)
#simulate a dataset with continuous data
dataset <- matrix( runif(1000 * 10, 1, 100), ncol = 10 )

#define a simulated class variable
target <- rnorm(1000)

a <- bs.reg(target, dataset, threshold = 0.05, test = "testIndRQ")
b <- bs.reg(target, dataset, threshold = 0.05, test = "testIndReg")
```

`Backward selection with generalised linear regression models`
Variable selection in generalised linear regression models with backward selection

---

Description

Variable selection in generalised linear regression models with backward selection

Usage

```
glm.bsreg(target, dataset, threshold = 0.05, wei = NULL, heavy = FALSE, robust = FALSE)
```

Arguments

`target`	The class variable. Provide either an integer, a numeric value, or a factor. It can also be a matrix with two columns for the case of binomial regression. In this case, the first column is the nubmer of successes and the second column is the number of trials. See also the Details.
`dataset`	The dataset; provide either a data frame or a matrix (columns = variables, rows = observations). In either case, only two cases are avaialble, either all data are continuous, or categorical.
`threshold`	Threshold (suitable values in (0, 1)) for assessing p-values significance. Default value is 0.05.
`wei`	A vector of weights to be used for weighted regression. The default value is NULL. It is not suggested when robust is set to TRUE.
`heavy`	A boolean variable specifying whether heavy computations are required or not. If for exmaple the dataset contains tens of thousands of rows, it is advised to used memory efficient GLMs and hence set this to TRUE.
`robust`	A boolean variable which indicates whether (TRUE) or not (FALSE) to use a robust version of the statistical test if it is available. It takes more time than a non robust version but it is suggested in case of outliers. Default value is FALSE. Currently only the linear regression option is supported.

Details

This functions currently implements only linear, binomial, binary logistic and Poisson regression. If the sample size is less than the number of variables a meesage will appear and no backward regression is performed.

Value

The output of the algorithm is S3 object including:

`runtime`	The run time of the algorithm. A numeric vector. The first element is the user time, the second element is the system time and the third element is the elapsed time.
`info`	A matrix with the variables and their latest test statistics and p-values.
`mat`	A matrix with the selected variables and their latest test statistic and p-value.
`ci_test`	The conditional independence test used.
`final`	The final regression model.

Author(s)

Michail Tsagris

R implementation and documentation: Michail Tsagris <mtsagris@csd.uoc.gr>

See Also

`fs.reg`, `lm.fsreg`, `bic.fsreg`, `bic.glm.fsreg`, `CondIndTests`, `MMPC`, `SES`

Examples

```
set.seed(123)

#simulate a dataset with continuous data
dataset <- matrix( runif(1000 * 10, 1, 100), ncol = 10 )

#define a simulated class variable
target <- rpois(1000, 10)
a <- glm.bsreg(target, dataset, threshold = 0.05)

target <- rnorm(1000)
b <- glm.bsreg(target, dataset, threshold = 0.05)

target <- rbinom(1000, 1, 0.6)
d <- glm.bsreg(target, dataset, threshold = 0.05)
```

`Beta regression` *Beta regression*

Description

Beta regression.

Usage

```
beta.mod(target, dataset, wei = NULL, xnew= NULL)
beta.reg(target, dataset, wei = NULL)
```

Arguments

`target`	The target (dependent) variable. It must be a numerical vector with integers.
`dataset`	The indendent variable(s). It can be a vector, a matrix or a dataframe with continuous only variables, a data frame with mixed or only categorical variables.
`wei`	A vector of weights to be used for weighted regression. The default value is NULL.
`xnew`	If you have new values for the predictor variables (dataset) whose target variable you want to predict insert them here. If you put the "dataset" or leave it NULL it will calculate the regression fitted values.

Details

The beta regression is fitted. The "beta.reg" is an internal wrapper function and is used for speed up purposes. It is not to be called directly by the user unless they know what they are doing.

Value

A list including:

`iters`	The iterations required until convergence.
`phi`	The estimated precision parameter.
`be`	The estimated coefficients of the model.
`loglik`	The log-likelihood of the regression model.

Author(s)

Michail Tsagris

R implementation and documentation: Michail Tsagris <mtsagris@csd.uoc.gr>

References

Ferrari S.L.P. and Cribari-Neto F. (2004). Beta Regression for Modelling Rates and Proportions. Journal of Applied Statistics, 31(7): 799-815.

See Also

`beta.regs`, `testIndBeta`, `reg.fit`, `ridge.reg`

Examples

```
y <- rbeta(500, 3, 5)
x <- matrix( rnorm(500 * 2), ncol = 2)
a1 <- beta.mod(y, x)
w <- runif(500)
a2 <- beta.mod(y, x, w)
```

`BIC based forward selection` *Variable selection in regression models with forward selection using BIC*

Description

Variable selection in regression models with forward selection using BIC

Usage

```
bic.fsreg(target, dataset, test = NULL, wei = NULL, tol = 2, robust = FALSE, ncores = 1)
```

Arguments

`target`	The class variable. Provide either a string, an integer, a numeric value, a vector, a factor, an ordered factor or a Surv object. See also Details.
`dataset`	The dataset; provide either a data frame or a matrix (columns = variables, rows = samples). The data can be either euclidean, categorical or both.
`test`	The regression model to use. Available options are "testIndReg" for normal linear regression, "testIndBeta" for beta regression, "censIndCR" or "censIndWR" for Cox proportional hazards and Weibull regression respectively, "testIndLogistic" for binomial, multinomial or ordinal regression, "testIndPois" for poisson regression, "testIndNB" for negative binomial regression, "testIndZIP" for zero inflated poisson regression, "testIndRQ" for quantile regression and "testIndSpeedglm" for linear, binary or poisson regression with large datasets (tens of thousands of observations). If you want to use multinomial or ordinal logistic regression, make sure your target is factor. See also `SES` and `CondIndTests` for the tests.
`wei`	A vector of weights to be used for weighted regression. The default value is NULL. It is not suggested when robust is set to TRUE.
`tol`	The difference bewtween two successive values of the stopping rule. By default this is is set to 2. If for example, the BIC difference between two succesive models is less than 2, the process stops and the last variable, even though significant does not enter the model.
`robust`	A boolean variable which indicates whether (TRUE) or not (FALSE) to use a robust version of the statistical test if it is available. It takes more time than a non robust version but it is suggested in case of outliers. Default value is FALSE.
`ncores`	How many cores to use. This plays an important role if you have tens of thousands of variables or really large sample sizes and tens of thousands of variables and a regression based test which requires numerical optimisation. In other cammmb it will not make a difference in the overall time (in fact it can be slower). The parallel computation is used in the first step of the algorithm, where univariate associations are examined, those take place in parallel. We have seen a reduction in time of 50% with 4 cores in comparison to 1 core. Note also, that the amount of reduction is not linear in the number of cores.

Details

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 or the binary logistic regression is used. If the target has two values only, binary logistic regression will be used.
- if target is a ordered factor, the ordinal regression is used.
- if target is a numerical vector or a matrix with at least two columns (multivariate) linear regression is used.
- if target is discrete numerical (counts), the poisson regression conditional independence test is used. If there are only two values, the binary logistic regression is to be used.
- if target is a Surv object, the Survival conditional independence test (Cox regression) is used.

Value

The output of the algorithm is S3 object including:

`mat`	A matrix with the variables and their latest test statistics and p-values.
`info`	A matrix with the selected variables, and the BIC of the model with that and all the previous variables.
`models`	The regression models, one at each step.
`final`	The final regression model.
`ci_test`	The conditional independence test used.
`runtime`	The run time of the algorithm. A numeric vector. The first element is the user time, the second element is the system time and the third element is the elapsed time.

Author(s)

Michail Tsagris

R implementation and documentation: Giorgos Athineou <athineou@csd.uoc.gr> Michail Tsagris <mtsagris@csd.uoc.gr>

References

Tsamardinos I., Aliferis C. F. and Statnikov, A. (2003). Time and sample efficient discovery of Markov blankets and direct causal relations. In Proceedings of the 9th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 673-678).

See Also

`glm.fsreg`, `lm.fsreg`, `bic.glm.fsreg`, `CondIndTests`, `MMPC`, `SES`

Examples

```
set.seed(123)

#simulate a dataset with continuous data
dataset <- matrix( runif(1000 * 20, 1, 100), ncol = 20 )

#define a simulated class variable
target <- 3 * dataset[, 10] + 2 * dataset[, 15] + 3 * dataset[, 20] + rnorm(1000, 0, 5)
a1 <- bic.fsreg(target, dataset, robust = FALSE, tol = 4, ncores = 1 )
a2 <- bic.fsreg(target, dataset, robust = TRUE, tol = 4, ncores = 1 )
a3 <- MMPC(target, dataset, robust= FALSE, ncores = 1)

target <- round(target)
b1 <- bic.fsreg(target, dataset, robust = FALSE, tol = 2, ncores = 1 )
b2 <- bic.fsreg(target, dataset, robust = TRUE, tol = 2, ncores = 1 )
# b3 <- MMPC(target, dataset, robust= FALSE, ncores = 1)  ## takes more time
```

BIC based forward selection with generalised linear models

Variable selection in generalised linear models with forward selection based on BIC

Description

Variable selection in generalised linear models with forward selection based on BIC

Usage

```
bic.glm.fsreg( target, dataset, wei = NULL, tol = 0, heavy = FALSE,
robust = FALSE, ncores = 1)
```

Arguments

`target`	The class variable. It can be either a vector with binary data (binomial regression), counts (poisson regression). If none of these is identified, linear regression is used.
`dataset`	The dataset; provide either a data frame or a matrix (columns = variables, rows = samples). These can be continous and or categorical.
`wei`	A vector of weights to be used for weighted regression. The default value is NULL. It is not suggested when robust is set to TRUE.
`tol`	The difference bewtween two successive values of BIC. By default this is is set to 2. If for example, the BIC difference between two succesive models is less than 2, the process stops and the last variable, even though significant does not enter the model.
`heavy`	A boolean variable specifying whether heavy computations are required or not. If for exmaple the dataset contains tens of thousands of rows, it is advised to used memory efficient GLMs and hence set this to TRUE.
`robust`	A boolean variable which indicates whether (TRUE) or not (FALSE) to use a robust version of the statistical test if it is available. It takes more time than a non robust version but it is suggested in case of outliers. Default value is FALSE and is currently supported only by linear regression
`ncores`	How many cores to use. This plays an important role if you have tens of thousands of variables or really large sample sizes and tens of thousands of variables and a regression based test which requires numerical optimisation. In other cammmb it will not make a difference in the overall time (in fact it can be slower). The parallel computation is used in the first step of the algorithm, where univariate associations are examined, those take place in parallel. We have seen a reduction in time of 50% with 4 cores in comparison to 1 core. Note also, that the amount of reduction is not linear in the number of cores.

Details

Forward selection via the BIC is implemented. A variable which results in a reduction of BIC will be included, until the reduction is below a threshold set by the user (argument "tol").

Value

The output of the algorithm is S3 object including:

`mat`	A matrix with the variables and their latest test statistics and p-values.
`info`	A matrix with the selected variables, and the BIC of the model with that and all the previous variables.
`models`	The regression models, one at each step.
`final`	The final regression model.
`ci_test`	The conditional independence test used.
`runtime`	The run time of the algorithm. A numeric vector. The first element is the user time, the second element is the system time and the third element is the elapsed time.

Author(s)

Michail Tsagris

R implementation and documentation: Giorgos Aathineou <athineou@csd.uoc.gr> Michail Tsagris <mtsagris@csd.uoc.gr>

See Also

`fs.reg`, `lm.fsreg`, `bic.fsreg`, `CondIndTests`, `MMPC`, `SES`

Examples

```
set.seed(123)

#simulate a dataset with continuous data
dataset <- matrix( runif(1000 * 50, 1, 100), ncol = 50 )

#define a simulated class variable
target <- 3 * dataset[, 10] + 2 * dataset[, 20] + 3 * dataset[, 30] + rnorm(1000, 0, 5)
a1 <- bic.glm.fsreg(target, dataset, robust = FALSE, tol = 2, ncores = 1 )
a2 <- bic.glm.fsreg( round(target), dataset, robust = FALSE, tol = 2, ncores = 1 )

y <- target   ;   me <- median(target)  ;   y[ y < me ] <- 0   ;   y[ y >= me ] <- 1
a3 <- bic.glm.fsreg( y, dataset, robust = FALSE, tol = 2, ncores = 1 )
```

`Check Markov equivalence of two DAGs`

Check Markov equivalence of two DAGs

Description

Check Markov equivalence of two DAGs.

Usage

```
equivdags(g1, g2)
```

Arguments

g1	The matrix of a DAG or a partially directed graph as produced from pc.or or any other algorithm.
g2	The matrix of a DAG or a partially directed graph as produced from pc.or or any other algorithm.

Details

Two DAGs are Markov equivalent if a) they have the same adjancencies (regardlsee of the mark, arrowhead, tail or nothing) and b) they have the same unshielded colliders.

Value

A list including:

apofasi	A boolean variable, TRUE of FALSE.
mes	A message specyfying the result, the dimensions of the adjacency matrices do not match for example, or the number of adjancencies is not the same, they do not share the same unshilded colliders, or they are Markov equivalent.

Author(s)

Michail Tsagris

R implementation and documentation: Giorgos Athineou <athineou@csd.uoc.gr> and Michail Tsagris <mtsagris@csd.uoc.gr>

References

Tsamardinos, Brown and Aliferis (2006). The max-min hill-climbing Bayesian network structure learning algorithm. Machine learning, 65(1), 31-78.

See Also

`pc.or`, `mmhc.skel`, `pc.con`

Examples

```
y <- rdag(1000, 10, 0.3)
tru <- y$G
x <- y$x
mod <- pc.con(x)

eg <- dag2eg(y$G) ## make it essential graph first
est <- pc.or(mod)$G

equivdags(est, tru)
```

CondInditional independence tests

MXM Conditional independence tests

Description

Currently the **MXM** package supports numerous tests for different types of target (dependent) and predictor (independent) variables. The target variable can be of continuous, discrete, categorical and of survival type. As for the predictor variables, they can be continuous, categorical or mixed.

The **testIndFisher** and the **gSquare** tests have two things in common. They do not use a model implicitly (i.e. estimate some beta coefficients), even though there is an underlying assumed one. Secondly they are pure tests of independence (again, with assumptions required).

As for the other tests, they share one thing in common. For all of them, two parametric models must be fit. The null model containing the conditioning set of variables alone and the alternative model containing the conditioning set and the candidate variable. The significance of the new variable is assessed via a log-likelihood ratio test with the appropriate degrees of freedom. All of these tests are summarized in the below table.

Target variable	**Predictor variables**	**Available tests**	**Short explanation**
Continuous	Continuous	testIndFisher (robust)	Partial correlation
Continuous	Continuous	testIndSpearman	Partial correlation
Continuous	Mixed	testIndReg (robust)	Linear regression
Continuous	Mixed	testIndRQ	Median regression
Proportions	Continuous	testIndFisher (robust) after logit transformation	Partial correlation
Proportions	Continuous	testIndSpearman after logit transformation	Partial correlation
Proportions	Mixed	testIndReg(robust) after logit transformation	Linear regression
Proportions	Mixed	testIndRQ after logit transformation	Median regression
Proportions	Mixed	testIndBeta	Beta regression
Non negative	Mixed	testIndIGreg	Inverse Gaussian regression
Non negative	Mixed	censIndWR	Weibull regression
Successes \& totals	Mixed	testIndBinom	Binomial regression
Discrete	Mixed	testIndPois	Poisson regression
Discrete	Mixed	testIndZIP	Zero Inflated Poisson regression
Discrete	Mixed	testIndNB	Negative binomial regression
Factor with two levels or binary	Mixed	testIndLogistic	Binary logistic regression
Factor with more than two levels (unordered)	Mixed	testIndLogistic	Multinomial logistic regression
Factor with more than two levels (ordered)	Mixed	testIndLogistic	Ordinal logistic regression

Categorical	Categorical	gSquare	G-squared test of independence
Categorical	Categorical	testIndLogistic	Multinomial logistic regression
Categorical	Categorical	testIndLogistic	Ordinal logistic regression
Continuous, proportions, binary or counts	Mixed	testIndSpeedglm	Linear, binary logistic or poisson gression
Survival	Mixed	censIndCR	Cox regression
Survival	Mixed	censIndWR	Weibull regression
Survival	Mixed	censIndER	Exponential regression
Case-control	Mixed	testIndClogit	Conditional logistic regression
Multivariate continuous	Mixed	testIndMVreg	Multivariate linear regression
Compositional data (no zeros)	Mixed	testIndMVreg after multivariate logit transformation	Multivariate linear regression
Longitudinal	Continuous	TestIndGLMM	(Generalised) linear mixed models

Details

These tests can be called by SES or individually by the user. In all regression cases, expect for the mixed models, there is an option for weights.

Tests

1. **testIndFisher**. This is a standard test of independence when both the target and the set of predictor variables are continuous (continuous-continuous). When the joint multivariate normality of all the variables is assumed, we know that if a correlation is zero this means that the two variables are independent. Moving in this spirit, when the partial correlation between the target variable and the new predictor variable conditioning on a set of (predictor) variables is zero, then we have evidence to say they are independent as well. An easy way to calculate the partial correlation between the target and a predictor variable conditioning on some other variables is to regress the both the target and the new variable on the conditioning set. The correlation coefficient of the residuals produced by the two regressions equals the partial correlation coefficient. If the robust option is selected, the two aforementioned regression models are fitted using M estimators (Marona et al., 2006). If the target variable consists of proportions or percentages (within the (0, 1) interval), the logit transformation is applied beforehand.
2. **testIndSpearman**. This is a non-parametric alternative to **testIndFisher** test. It is a bit slower than its competitor, yet very fast and suggested when normality assumption breaks down or outliers are present. In fact, within SES, what happens is that the ranks of the target and of the dataset (predictor variables) are computed and the **testIndSpearman** is aplied. This is faster than applying Fisher with M estimators as described above. If the target variable consists of proportions or percentages (within the (0, 1) interval), the logit transformation is applied beforehand.
3. **testIndReg**. In the case of target-predictors being continuous-mixed or continuous-categorical, the suggested test is via the standard linear regression. In this case, two linear regression models are fitted. One with the conditioning set only and one with the conditioning set plus the new variable. The significance of the new variable is assessed via the F test, which calculates

the residual sum of squares of the two models. The reason for the F test is because the new variable may be categorical and in this case the t test cannot be used. It makes sense to say, that this test can be used instead of the **testIndFisher**, but it will be slower. If the robust option is selected, the two models are fitted using M estimators (Marona et al. 2006). If the target variable consists of proportions or percentages (within the (0, 1) interval), the logit transformation is applied beforehand.

4. **testIndRQ**. An alternative to **testIndReg** for the case of continuous-mixed (or continuous-continuous) variables is the **testIndRQ**. Instead of fitting two linear regression models, which model the expected value, one can choose to model the median of the distribution (Koenker, 2005). The significance of the new variable is assessed via a rank based test calibrated with an F distribution (Gutenbrunner et al., 1993). The reason for this is that we performed simulation studies and saw that this type of test attains the type I error in contrast to the log-likelihood ratio test. The benefit of this regression is that it is robust, in contrast to the classical linear regression. If the target variable consists of proportions or percentages (within the (0, 1) interval), the logit transformation is applied beforehand.

5. **testIndBeta**. When the target is proportion (or percentage, i.e., between 0 and 1, not inclusive) the user can fit a regression model assuming a beta distribution. The predictor variables can be either continuous, categorical or mixed. The procedure is the same as in the **testIndReg** case.

6. **Alternatives to testIndBeta**. Instead of **testIndBeta** the user has the option to choose all the previous to that mentioned tests by transforming the target variable with the logit transformation. In this way, the support of the target becomes the whole of R^d and then depending on the type of the predictors and whether a robust approach is required or not, there is a variety of alternative to beta regression tests.

7. **testIndIGreg**. When you have non negative data, i.e. the target variable takes positive values (including 0), a suggested regression is based on the the inverse gaussian distribution. The link function is not the inverse of the square root as expected, but the logarithm. This is to ensure that the fitted values will be always be non negative. The predictor variables can be either continuous, categorical or mixed. The significance between the two models is assessed via the log-likelihood ratio test. Alternatively, the user can use the Weibull regression (**censIndWR**).

8. **testIndPois**. When the target is discrete, and in specific count data, the default test is via the Poisson regression. The predictor variables can be either continuous, categorical or mixed. The procedure is the same as in all the previously regression model based tests, i.e. the log-likelihood ratio test is used to assess the conditional independence of the variable of interest.

9. **testIndNB**. As an alternative to the Poisson regression, we have included the Negative binomial regression to capture cases of overdispersion. The predictor variables can be either continuous, categorical or mixed.

10. **testIndZIP**. When the number of zeros is more than expected under a Poisson model, the zero inflated poisson regression is to be employed. The predictor variables can be either continuous, categorical or mixed.

11. **testIndLogistic** (Binomial). When the target is categorical with only two outcomes, success or failure for example, then a binary logistic regression is to be used. Whether regression or classification is the task of interest, this method is applicable. The advantage of this over a linear or quadratic discriminant analysis is that it allows for categorical predictor variables as well and for mixed types of predictors.

12. **testIndLogistic** (Un-ordered multinomial). If the target has more than two outcomes, but it is of nominal type, there is no ordering of the outcomes, multinomial logistic regression will be employed. Again, this regression is suitable for classification purposes as well and it to allows for categorical predictor variables.
13. **testIndLogistic** (Ordered multinomial). This is a special case of multinomial regression, in which case the outcomes have an ordering, such as **not satisfied**, **neutral**, **satisfied**. The appropriate method is ordinal logistic regression.
14. **testIndBinom**. When the target variable is a matrix of two columns, where the first one is the number of successes and the second one is the number of trials, binomial regression is to be used.
15. **testIndSpeedglm**. If you have a few tens of thousands of observations, the default functions for linear, binary logistic and poisson regression will be slow causing the computer to jamm. For this reason, memory efficient handling regressions should be used.
16. **gSquare**. If all variables, both the target and predictors are categorical the default test is the G-square test of independence. It is similar to the chi-squared test of independence, but instead of using the chi-squared metric between the observed and estimated frequencies in contingency tables, the Kullback-Leibler divergence of the observed from the estimated frequencies is used. The asymptotic distribution of the test statistic is a chi-squared distribution on some appropriate degrees of freedom. The target variable can be either ordered or unordered with two or more outcomes.
17. **Alternatives to gSquare**. An alternative to the **gSquare** test is the **testIndLogistic**. Depending on the nature of the target, binary, un-ordered multinomial or ordered multinomial the appropriate regression model is fitted.
18. **censIndCR**. For the case of time-to-event data, a Cox regression model is employed. The predictor variables can be either continuous, categorical or mixed. Again, the log-likelihood ratio test is used to assess the significance of the new variable.
19. **censIndWR**. A second model for the case of time-to-event data, a Weibull regression model is employed. The predictor variables can be either continuous, categorical or mixed. Again, the log-likelihood ratio test is used to assess the significance of the new variable. Unlike the semi-parametric Cox model, the Weibull model is fully parametric.
20. **censIndER**. A third model for the case of time-to-event data, an exponential regression model is employed. The predictor variables can be either continuous, categorical or mixed. Again, the log-likelihood ratio test is used to assess the significance of the new variable. This is a special case of the Weibull model.
21. **testIndClogit**. When the data come from a case-control study, the suitable test is via conditional logistic regression.
22. **testIndMVReg**. In the case of multivariate continuous targets, the suggested test is via a multivariate linear regression. The target variable can be compositional data as well. These are positive data, whose vectors sum to 1. They can sum to any constant, as long as it the same, but for convenience reasons we assume that they are normalised to sum to 1. In this case the additive log-ratio transformation (multivariate logit transformation) is applied beforehand.
23. **testIndGLMM**. In the case of a longitudinal or clustered targets (continuous, proportions, binary or counts), the suggested test is via a (generalised) linear mixed model.

Author(s)

Michail Tsagris <mtsagris@csd.uoc.gr>

References

Aitchison J. (1986). The Statistical Analysis of Compositional Data, Chapman & Hall; reprinted in 2003, with additional material, by The Blackburn Press.

Brown P.J. (1994). Measurement, Regression and Calibration. Oxford Science Publications.

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

Draper, N.R. and Smith H. (1988). Applied regression analysis. New York, Wiley, 3rd edition.

Fieller E.C. and Pearson E.S. (1961). Tests for rank correlation coefficients: II. Biometrika, 48(1 & 2): 29-40.

Ferrari S.L.P. and Cribari-Neto F. (2004). Beta Regression for Modelling Rates and Proportions. Journal of Applied Statistics, 31(7): 799-815.

Gail, M.H., Jay H.L., and Lawrence V.R. (1981). Likelihood calculations for matched case-control studies and survival studies with tied death times. Biometrika 68(3): 703-707.

Gutenbrunner C., Jureckova J., Koenker R. and Portnoy S. (1993). Tests of Linear Hypothesis based on Regression Rank Scores, Journal of NonParametric Statistics 2, 307-331.

Hoerl A.E. and Kennard R.W. (1970). Ridge regression: Biased estimation for nonorthogonal problems. Technometrics, 12(1): 55-67.

Joseph M.H. (2011). Negative Binomial Regression. Cambridge University Press, 2nd edition.

Koenker R.W. (2005). Quantile Regression. Cambridge University Press.

Lagani V., Kortas G. and Tsamardinos I. (2013). Biomarker signature identification in "omics" with multiclass outcome. Computational and Structural Biotechnology Journal, 6(7): 1-7.

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

Lambert D. (1992). Zero-inflated Poisson regression, with an application to defects in manufacturing. Technometrics 34(1)1: 1-14.

Mardia K.V., Kent J.T. and Bibby J.M. (1979). Multivariate Analysis. Academic Press, New York, USA.

Maronna R.D. Yohai M.V. (2006). Robust Statistics, Theory and Methods. Wiley.

McCullagh P. and Nelder J.A. (1989). Generalized linear models. CRC press, USA, 2nd edition.

Pinheiro J., and D. Bates. Mixed-effects models in S and S-PLUS. Springer Science \& Business Media, 2006.

Scholz, F. W. (2001). Maximum likelihood estimation for type I censored Weibull data including covariates. ISSTECH-96-022, Boeing Information & Support Services.

Smith, R. L. (1991). Weibull regression models for reliability data. Reliability Engineering & System Safety, 34(1), 55-76.

Spirtes P., Glymour C. and Scheines R. (2001). Causation, Prediction, and Search. The MIT Press, Cambridge, MA, USA, 3nd edition.

`Conditional independence test for binary, categorical or ordinal data` *Conditional independence test 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 the aproprirate degrees of freedom on the difference between the deviances of the two models.

Usage

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

Arguments

`target`	A numeric vector containing the values of the target variable.
`dataset`	A numeric matrix or data frame, in case of categorical predictors (factors), 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.
`wei`	A vector of weights to be used for weighted regression. The default value is NULL.
`dataInfo`	A 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)
`robust`	A boolean variable which indicates whether (TRUE) or not (FALSE) to use a robustified version of the logistic regressions available here. Currently it is not available.

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. For all the available conditional independence tests that are currently included on the package, please see "?CondIndTests".

Value

A list including:

`pvalue`	A numeric value that represents the logarithm of 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 multinom (package nnet) for multinomial logistic regression, the function clm (package ordinal) for ordinal logit regression and the function glm (package stats) for binomial regression.

Author(s)

Vincenzo Lagani and Ioannis Tsamardinos

R implementation and documentation: Vincenzo Lagani <vlagani@csd.uoc.gr>, Giorgos Athineou <athineou@csd.uoc.gr> and Michail Tsagris <mtsagris@csd.uoc.gr>

References

Hampel F. R., Ronchetti E. M., Rousseeuw P. J., and Stahel W. A. (1986). Robust statistics: the approach based on influence functions. John Wiley & Sons.

Vincenzo Lagani, George Kortas and Ioannis Tsamardinos (2013), Biomarker signature identification in "omics" with multiclass outcome. Computational and Structural Biotechnology Journal, 6(7):1-7.

McCullagh, Peter, and John A. Nelder. Generalized linear models. CRC press, USA, 2nd edition, 1989.

See Also

`SES`, `testIndSpeedglm`, `gSquare`, `CondIndTests`

Examples

```
#require(nnet)
#require(ordinal)

#simulate a dataset with categorical data
dataset_m <- matrix( sample(c(0, 1, 2), 50 * 100, replace = TRUE), ncol = 50)
#initialize categorical target
target_m <- dataset_m[, 50]
#remove target from the dataset
dataset_m <- dataset_m[, -50]

  #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

  #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

  #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(sample(c(0,1),50 * 60, replace = TRUE), ncol = 50)
#initialize binary target
target_b <- dataset_b[, 50]
#remove target from the dataset
dataset_b <- dataset_b[, -50]

  #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

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

`Conditional independence test for case control data`

Conditional independence test based on conditional logistic regression for case control studies

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 conditional logistic regression model based on the conditioning set CS against a model whose regressors are both X and CS. The comparison is performed through a chi-square test with the appropriate degrees of freedom on the difference between the deviances of the two models. This is suitable for a case control design

Usage

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

Arguments

`target`	A matrix with two columns, the first one must be 0 and 1, standing for 0 = control and 1 = case. The second column is the id of the patients. A numerical variable, for example c(1,2,3,4,5,6,7,1,2,3,4,5,6,7).
`dataset`	A numeric matrix or a data.frame in case of categorical predictors (factors), 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.
`wei`	A vector of weights to be used for weighted regression. The default value is NULL and it should stay NULL as weights are ignored the conditional logistic regression. See the **survival** package for more information about this type of regression.
`dataInfo`	A 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.
`robust`	A boolean variable which indicates whether (TRUE) or not (FALSE) to use a robustified version of Beta regression. Currently it is not available for this test.

Details

If hash = TRUE, testIndClogit 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. For all the available conditional independence tests that are currently included on the package, please see "?CondIndTests".

This is for case control studies. The logliklihood for a conditional logistic regresson model equals the loglikelihood from a Cox model with a particular data structure. When a well tested Cox model routine is available many packages use this "trick" rather than writing a new software routine from scratch, and this is what the "clogit" function in the "survival" package does. In detail, a stratified Cox model with each case/control group assigned to its own stratum, time set to a constant, status of 1=case 0=control, and using the exact partial likelihood has the same likelihood formula as a conditional logistic regression. The "clogit" routine creates the necessary dummy variable of times (all 1) and the strata, then calls the function "coxph".

Value

A list including:

`pvalue`	A numeric value that represents the logarithm of the generated p-value due to the conditional logistic regression (see reference below).
`stat`	A numeric value that represents the generated statistic due to the conditional logistic regression (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, Ioannis Tsamardinos, Giorgos Athineou and Michail Tsagris

R implementation and documentation: Giorgos Athineou <athineou@csd.uoc.gr>, Vincenzo Lagani <vlagani@csd.uoc.gr> and Michail Tsagris <mtsagris@csd.uoc.gr>

References

Michell H. Gail, Jay H. Lubin and Lawrence V. Rubinstein (1980). Likelihood calculations for matched case-control studies and survival studies with tied death times. Biometrika 68:703-707.

See Also

`SES`, `testIndLogistic`, `censIndCR`, `censIndWR`

Examples

```
#simulate a dataset with continuous data
dataset <- matrix(rnorm(300 * 100), nrow = 300 )
#the target feature is the last column of the dataset as a vector
case <- rbinom(300, 1, 0.6)
```

```
ina <- which(case==1)
ina <- sample(ina, 100)
case[-ina] = 0
id <- rep(1:100,3)
target <- cbind(case, id)

results <- testIndClogit(target, dataset, xIndex = 44, csIndex = 60)
results

#run the SES algorithm using the testIndClogit conditional independence test
a1 <- SES(target, dataset, max_k = 3, threshold = 0.05, test = "testIndClogit");
a2 <- MMPC(target, dataset, max_k = 3, threshold = 0.05, test = "testIndClogit");
# print summary of the SES output
summary(a1);
# plot the SES output
# plot(a1, mode = "all");
```

`Conditional independence test for continuous, binary and count data with thousands of samples`

Conditional independence test for continuous, binary and discrete (counts) variables with thousands of observations

---

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 the aproprirate degrees of freedom on the difference between the deviances of the two models.

Usage

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

Arguments

`target`	A numeric vector containing the values of the target variable. It can be either continuous or percentages (values within 0 and 1), binary or discrete (counts).
`dataset`	A numeric matrix or data frame, in case of categorical predictors (factors), 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.
`wei`	A vector of weights to be used for weighted regression. The default value is NULL.

`dataInfo`	A 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 (discrete target)
`robust`	A boolean variable which indicates whether (TRUE) or not (FALSE) to use a robustified version of the logistic regressions available here. Currently it is not available for these cases.

Details

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

- dataInfo$target_type = "normal" (continuous target)
- dataInfo$target_type = "binary" (binary target)
- dataInfo$target_type = "discrete" (discrete target)

If hash = TRUE, testIndSpeedglm 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.

This test is designed for large sample sized data, tens and hundreds of thousands and it works for linear, logistic and poisson regression. The classical `lm` and `glm` functions will use too much memory when many observations are available. The package "speedglm" handles such data more efficiently. You can try and see, in the first case the computer will jam, whereas in the second it will not. Hence, this test is to be used in these cases only. We have not set a threshold on the sample size, so that the algorithm decides whether to shift to speedglm or not, because this depends upon the user's computing fascilities. When there are up to $20,000$ observations, the built-in function `lm` is faster, but when $n = 30,000$, the `speedlm` is more than twice as fast.

For all the available conditional independence tests that are currently included on the package, please see "?CondIndTests".

Value

A list including:

`pvalue`	A numeric value that represents the logarithm of 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.

Author(s)

Vincenzo Lagani and Ioannis Tsamardinos

R implementation and documentation: Vincenzo Lagani <vlagani@csd.uoc.gr>, Giorgos Athineou <athineou@csd.uoc.gr> and Michail Tsagris <mtsagris@csd.uoc.gr>

References

McCullagh, Peter, and John A. Nelder. Generalized linear models. CRC press, USA, 2nd edition, 1989.

See Also

`SES`, `testIndLogistic`, `testIndReg`, `testIndPois`, `CondIndTests`

Examples

```
dataset <- matrix(runif(40000 * 10, 1, 50), ncol = 10 )
#the target feature is the last column of the dataset as a vector
target <- rpois(40000, 10)
system.time( testIndPois(target, dataset, xIndex = 1, csIndex = 2) )
system.time( testIndSpeedglm(target, dataset, xIndex = 1, csIndex = 2) )
```

Conditional independence test for longitudinal and clustered data

Linear mixed models conditional independence test for longitudinal 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 linear model based on the conditioning set CS against a model with both X and CS. The comparison is performed through an F test the appropriate degrees of freedom on the difference between the deviances of the two models. This test accepts a longitudinal target and longitudinal, categorical, continuous or mixed data as predictor variables.

Usage

```
testIndGLMM(target, reps = NULL, group, dataset, xIndex, csIndex, wei = NULL,
dataInfo = NULL, univariateModels = NULL, hash = FALSE, stat_hash = NULL,
pvalue_hash=NULL, target_type = 0, slopes = FALSE)
```

Arguments

`target`	A numeric vector containing the values of the target variable. If the values are proportions or percentages, i.e. strictly within 0 and 1 they are mapped into R using log(target/(1 - target)). In both cases a linear mixed model is applied. It can also be a binary variable (binary logistic regression) or a discrete, counts (Poisson regression), thus fitting generalised linear mixed models.
`reps`	A numeric vector containing the time points of the subjects. It's length is equal to the length of the target variable. If you have clustered data, leave this NULL.
`group`	A numeric vector containing the subjects or groups. It must be of the same legnth as target.
`wei`	A vector of weights to be used for weighted regression. The default value is NULL.
`dataset`	A numeric matrix or data frame, in case of categorical predictors (factors), 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`	A 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.
`target_type`	The type of the target variable. It is set to 0 by default. 1 for continuous variable (normal), 2 for binary variable (binomial) and 3 for discrete variable (Poisson).
`slopes`	A boolean variable which indicates whether (TRUE) to or not (FALSE) random slopes in the time effect as well. By deault random intercepts are considered.

Details

If hash = TRUE, testIndGLMM 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.temp 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. For all the available conditional independence tests that are currently included on the package, please see "?CondIndTests".

This test is for longitudinal and clustered data. Bear in mind that the time effect, for the longitudinal data case, is linear. It could be of higer order as well, but this would be a hyper-parameter, increasing the complexity of the models to be tested.

Value

A list including:

`pvalue`	A numeric value that represents the logarithm of the generated p-value due to the (generalised) linear mixed model (see reference below).
`stat`	A numeric value that represents the generated statistic due to the (generalised) linear mixed model (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, Ioannis Tsamardinos, Michail Tsagris and Giorgos Athineou

R implementation and documentation: Giorgos Athineou <athineou@csd.gr>, Vincenzo Lagani <vlagani@csd.uoc.gr> and Michail Tsagris <mtsagris@csd.uoc.gr>

References

Pinheiro, Jose, and Douglas Bates. Mixed-effects models in S and S-PLUS. Springer Science \& Business Media, 2006.

See Also

SES.temporal, MMPC.temporal, CondIndTests

Examples

```
#data(sleepstudy)
#attach(sleepstudy)
#target <- Reaction
#x <- matrix(rnorm(180 * 10),ncol = 10) ## unrelated preidctor variables
#testIndGLMM(target, Days, Subject, x, 1,0,target_type = 1)
```

Conditional independence test for proportions/percentages

Beta regression conditional independence test for proportions/percentage class dependent variables and mixed predictors

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 Beta regression 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 the appropriate degrees of freedom on the difference between the deviances of the two models.

Usage

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

Arguments

target	A numeric vector containing the values of the target variable. They must be percentages or proportions, i.e. within the (0, 1) interval. Currently 0 and/or 1 values are not allowed.
dataset	A numeric matrix or data frame, in case of categorical predictors (factors), 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.

`wei`	A vector of weights to be used for weighted regression. The default value is NULL.
`dataInfo`	A 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.
`robust`	A boolean variable which indicates whether (TRUE) or not (FALSE) to use a robustified version of Beta regression. Currently it is not available for this test.

Details

If hash = TRUE, testIndBeta 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. For all the available conditional independence tests that are currently included on the package, please see "?CondIndTests". An alternative regression to this is "testIndReg" and "testIndRQ" .In these two latter cases, the logit transformation is first applied to the target variable.

Value

A list including:

`pvalue`	A numeric value that represents the logarithm of the generated p-value due to Beta regression (see reference below).
`stat`	A numeric value that represents the generated statistic due to Beta regression (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, Ioannis Tsamardinos, Michail Tsagris and Giorgos Athineou

R implementation and documentation: Giorgos Athineou <athineou@csd.uoc.gr>, Vincenzo Lagani <vlagani@csd.uoc.gr> and Michail Tsagris <mtsagris@csd.uoc.gr>

References

Ferrari S.L.P. and Cribari-Neto F. (2004). Beta Regression for Modelling Rates and Proportions. Journal of Applied Statistics, 31(7): 799-815.

See Also

`SES`, `testIndReg`, `testIndRQ`, `testIndFisher`, `CondIndTests`

Examples

```
#simulate a dataset with continuous data
dataset <- matrix(runif(100 * 20, 1, 1000), ncol = 20 )
#the target feature is the last column of the dataset as a vector
target <- dataset[, 20]
dataset <- dataset[, -20]
target <- target / (max(target) + 2 )

results <- testIndBeta(target, dataset, xIndex = 14, csIndex = 9)
results

#run the SES algorithm using the testIndBeta conditional independence test
sesObject <- SES(target, dataset, max_k = 3, threshold = 0.05, test = "testIndBeta");
#print summary of the SES output
summary(sesObject);
#plot the SES output
# plot(sesObject, mode = "all");
```

`Conditional independence tests for continous univariate and multivariate data`
Linear (and non-linear) regression conditional independence test for continous univariate and multivariate response 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 linear regression model based on the conditioning set CS against a model whose regressor are both X and CS. The comparison is performed through an F test the appropriate degrees of freedom on the difference between the deviances of the two models.

Usage

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

testIndRQ(target, dataset, xIndex, csIndex, wei = NULL, dataInfo = NULL,
univariateModels = NULL, hash = FALSE, stat_hash = NULL, pvalue_hash = NULL,
robust = FALSE)

testIndMVreg(target, dataset, xIndex, csIndex, wei = NULL, dataInfo = NULL,
univariateModels = NULL, hash = FALSE, stat_hash = NULL, pvalue_hash = NULL,
robust = FALSE)

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

Arguments

`target`	A numeric vector containing the values of the target variable. If the values are proportions or percentages, i.e. strictly within 0 and 1 they are mapped into R using log(target/(1 - target)). In the case of testIndMVreg, the same takes place true. See details for more information.
`dataset`	A numeric matrix or data frame, in case of categorical predictors (factors), 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.
`wei`	A vector of weights to be used for weighted regression. The default value is NULL. We suggest not to use weights if you choose testIndReg and robust = TRUE (robust regression via M estimation)
`dataInfo`	A 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.

`robust`	A boolean variable which indicates whether (TRUE) or not (FALSE) to use a robust regression via MM-estimation available from `rlm` in the package "MASS". A robust F test is also performed.It takes more time than non robust version but it is suggested in case of outliers. Default value is FALSE. This is only used in testIndReg. Quantile regression is robust by default and for multivariate regression this has not been incorporated yet.

Details

If hash = TRUE, all three tests require 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.

TestIndReg offers linear and robust linear (via M estimation) regression.

TestIndRQ offers quantile (median) regression as a robust alternative to linear regression.

In both cases, if the dependent variable consists of proportions (values between 0 and 1) the logit transformation is applied and the tests are applied then.

testIndMVreg is for multivariate continuous response variables. Compositional data are positive multivariate data and each vector (observation) sums to the same constant, usually taken 1 for convenience. A check is performed and if such data are found, the additive log-ratio (multivariate logit) transformation (Aitchison, 1986) is applied beforehand. Zeros are not allowed. TestIndIGreg is for non negative data. It fits an inverse gaussian distribution with a log link. Or you cound see it as a non linear Gaussian model where the conditional mean is related with the covariate(s) via an exponential function.

For all the available conditional independence tests that are currently included on the package, please see "?CondIndTests".

Value

A list including:

`pvalue`	A numeric value that represents the logarithm of the generated p-value due to linear regression (see reference below).
`stat`	A numeric value that represents the generated statistic due to linear regression (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, Ioannis Tsamardinos, Michail Tsagris and Giorgos Athineou

R implementation and documentation: Giorgos Athineou <athineou@csd.gr>, Vincenzo Lagani <vlagani@csd.uoc.gr> and Michail Tsagris <mtsagris@csd.uoc.gr>

References

Draper, N.R. and Smith H. (1988). Applied regression analysis. New York, Wiley, 3rd edition.

Hampel F. R., Ronchetti E. M., Rousseeuw P. J., and Stahel W. A. (1986). Robust statistics: the approach based on influence functions. John Wiley & Sons.

Koenker R.W. (2005). Quantile regression. New York, Cambridge University Press.

Mardia, Kanti, John T. Kent and John M. Bibby. Multivariate analysis. Academic press, 1979.

John Aitchison. The Statistical Analysis of Compositional Data, Chapman & Hall; reprinted in 2003, with additional material, by The Blackburn Press.

See Also

`testIndSpeedglm`, `testIndRQ`, `testIndFisher`, `testIndSpearman`, `CondIndTests`

Examples

```
#simulate a dataset with continuous data
dataset <- matrix(runif(100 * 100, 1, 100), ncol = 100 )
#the target feature is the last column of the dataset as a vector
target <- dataset[, 100]
dataset <- dataset[, -100]

testIndReg(target, dataset, xIndex = 44, csIndex = 50)
testIndReg(target, dataset, xIndex = 44, csIndex = 50, robust = TRUE)
testIndRQ(target, dataset, xIndex = 44, csIndex = 50)
testIndIGreg(target, dataset, xIndex = 44, csIndex = 50)

#define class variable (here tha last column of the dataset)
#run the SES algorithm using the testIndReg conditional independence test
sesObject <- SES(target, dataset, max_k = 3, threshold = 0.05, test = "testIndReg");
sesObject2 <- SES(target, dataset, max_k = 3, threshold = 0.05, test = "testIndRQ");
#print summary of the SES output

summary(sesObject);
summary(sesObject2);
# plot the SES output
# plot(sesObject, mode = "all");
```

Conditional independence tests for count data

Regression conditional independence test for discrete (counts) class dependent 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 Poisson regression 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 the appropriate degrees of freedom on the difference between the deviances of the two models. The models supported here are poisson, zero inlftaed poisson and negative binomial.

Usage

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

testIndNB(target, dataset, xIndex, csIndex, wei = NULL, dataInfo = NULL,
univariateModels = NULL, hash = FALSE, stat_hash = NULL, pvalue_hash = NULL,
robust = FALSE)

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

Arguments

`target`	A numeric vector containing the values of the target variable.
`dataset`	A numeric matrix or data frame, in case of categorical predictors (factors), 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.
`wei`	A vector of weights to be used for weighted regression. The default value is NULL.
`dataInfo`	A 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.
`robust`	A boolean variable which indicates whether (TRUE) or not (FALSE) to use a robust version of the statistical test if it is available. It takes more time than non robust version but it is suggested in case of outliers. Default value is FALSE as it is currently nor supported.

Details

If hash = TRUE, all three tests require 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. For all the available conditional independence tests that are currently included on the package, please see "?CondIndTests".

If you have overdispersion, the variance is higher than the mean, a negative binomial is to be used. If you have more zeros than expected under a Poisson model, not overdispersion, then zero inlfated Poisson is to be used. Bear in mind that if you have a small number of zeros, there is no reason to use this model. If for example you have count data but no, or 1 zeros, this will not work.

Value

A list including:

`pvalue`	A numeric value that represents the logarithm of the generated p-value due to the count data regression (see references below).
`stat`	A numeric value that represents the generated statistic due to Poisson regression(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, Ioannis Tsamardinos, Michail Tsagris and Giorgos Athineou

R implementation and documentation: Giorgos Athineou <athineou@csd.uoc.gr>, Vincenzo Lagani <vlagani@csd.uoc.gr> and Michail Tsagris <mtsagris@csd.uoc.gr>

References

McCullagh P., and Nelder J.A. (1989). Generalized linear models. CRC press, USA, 2nd edition.

Lambert D. (1992). Zero-inflated Poisson regression, with an application to defects in manufacturing. Technometrics, 34(1):1-14.

Joseph M.H. (2011). Negative Binomial Regression. Cambridge University Press, 2nd edition.

See Also

`testIndSpeedglm`, `testIndNB`, `testIndZIP`, `gSquare`, `CondIndTests`

Examples

```
#simulate a dataset with continuous data
dataset <- matrix(runif(400 * 50, 1, 50), ncol = 50 )
#the target feature is the last column of the dataset as a vector
target <- rpois(400, 10)
results <- testIndPois(target, dataset, xIndex = 24, csIndex = 10)
results

#run the SES algorithm using the testIndPois conditional independence test
sesObject <- SES(target, dataset, max_k = 3, threshold = 0.05, test = "testIndPois");
sesObject2 <- SES(target, dataset, max_k = 3, threshold = 0.05, test = "testIndNB");
#print summary of the SES output
summary(sesObject);
# plot the SES output
# plot(sesObject, mode = "all");
```

`Conditional independence tests for sucess rates`
Binomial regression conditional independence test for success rates (binomial)

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 binomial logistic regression 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 the appropriate degrees of freedom on the difference between the deviances of the two models.

Usage

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

Arguments

`target`	A matrix with two two columns, the first one is the number of successes, the cases (integer) and the second one is the totals (integers).
`dataset`	A numeric matrix or data frame, in case of categorical predictors (factors), 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.
`wei`	A vector of weights to be used for weighted regression. The default value is NULL and should stay as is, since the totals (second column of the target) is used as weights.
`dataInfo`	A 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.
`robust`	A boolean variable which indicates whether (TRUE) or not (FALSE) to use a robust version of the statistical test if it is available. It takes more time than non robust version but it is suggested in case of outliers. Default value is FALSE as it is currently nor supported.

Details

If hash = TRUE, all three tests require 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. For all the available conditional independence tests that are currently included on the package, please see "?CondIndTests".

If you have overdispersion, the variance is higher than the mean, a negative binomial is to be used. If you have more zeros than expected under a Poisson model, not overdispersion, then zero inlfated Poisson is to be used. This is in fact a logistic reression where the target is the ratio of successes divided by the totals and the weights are the totals.

Value

A list including:

`pvalue`	A numeric value that represents the logarithm of the generated p-value due to the count data regression (see references below).
`stat`	A numeric value that represents the generated statistic due to Poisson regression(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, Ioannis Tsamardinos, Giorgos Athineou and Michail Tsagris.

R implementation and documentation: Giorgos Athineou <athineou@csd.uoc.gr>, Vincenzo Lagani <vlagani@csd.uoc.gr> and Michail Tsagris <mtsagris@csd.uoc.gr>

References

McCullagh P., and Nelder J.A. (1989). Generalized linear models. CRC press, USA, 2nd edition.

See Also

`testIndLogistic`, `testIndBeta`, `testIndReg`, `CondIndTests`

Examples

```
#simulate a dataset with continuous data
dataset <- matrix(runif(400 * 50, 1, 50), ncol = 50 )
#the target feature is the last column of the dataset as a vector
y <- rbinom(400, 10, 0.6)
N <- runif(400, 11, 20)
target <- cbind(y/N, N)
results <- testIndBinom(target, dataset, xIndex = 24, csIndex = 10)
results

#run the SES algorithm using the testIndPois conditional independence test
a1 <- SES(target, dataset, max_k = 3, threshold = 0.05, test = "testIndBinom");
a2 <- MMPC(target, dataset, max_k = 3, threshold = 0.05, test = "testIndBinom");
```

`Conditional independence tests for survival data`
Conditional independence test for survival data

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 can based on the Cox (semi-parametric) regression or on the Weibull (parametric) regression.

Usage

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

censIndWR(target, dataset, xIndex, csIndex, wei = NULL, dataInfo = NULL,
univariateModels = NULL, hash = FALSE, stat_hash = NULL, pvalue_hash = NULL,
robust = FALSE)

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

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 matrix or data frame, in case of categorical predictors (factors), 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.
`wei`	A vector of weights to be used for weighted regression. The default value is NULL.
`dataInfo`	A 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.
`robust`	A boolean variable which indicates whether (TRUE) or not (FALSE) to use a robustified version of Cox regression. Currently the robust version is not available for this test. Note, that Cox and Weibull regressions offer robust (sandwich) estimation of the standard error of the coefficients, but not robust estimation of the parameters.

Details

The censIndCR implies the Cox (semiparametric) regression, the censIndWR the Weibull (parametric) regression and the censIndER the exponential (parametric) regression, which is a special case of the Weibull regression (when shape parameter is 1).

If hash = TRUE, censIndCR, censIndWR and censIndER require 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.

For all the available conditional independence tests that are currently included on the package, please see "?CondIndTests".

Value

A list including:

`pvalue`	A numeric value that represents the logarithm of 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@csd.uoc.gr>, Giorgos Athineou <athineou@csd.uoc.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.

Scholz, F. W. (2001). Maximum likelihood estimation for type I censored Weibull data including covariates. ISSTECH-96-022, Boeing Information & Support Services.

Smith, R. L. (1991). Weibull regression models for reliability data. Reliability Engineering & System Safety, 34(1), 55-76.

See Also

`SES`, `censIndWR`, `testIndFisher`, `gSquare`, `testIndLogistic`, `Surv`, `anova`, `CondIndTests`

Examples

```
#create a survival simulated dataset
dataset <- matrix(runif(1000 * 20, 1, 100), nrow = 1000 , ncol = 20)
dataset <- as.data.frame(dataset);
timeToEvent <- numeric(1000)
event <- numeric(1000)
ca <- numeric(1000)
for(i in 1:1000) {
 timeToEvent[i] <- dataset[i, 1] + 0.5 * dataset[i, 10] + 2 * dataset[i, 15] + runif(1, 0, 1);
  event[i] <- sample( c(0, 1), 1)
  ca[i] <- runif(1, 0, timeToEvent[i]-0.5)
  if(event[i] == 0) {
    timeToEvent[i] = timeToEvent[i] - ca[i]
  }
}

require(survival, quietly = TRUE)

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

  #run the censIndCR   conditional independence test
  res <- censIndCR( target, dataset, xIndex = 12, csIndex = c(5, 7, 4) )
  res

  #run the SES algorithm using the censIndCR conditional independence
  #test for the survival class variable
  ses1 <- SES(target, dataset, max_k = 1, threshold = 0.05, test = "censIndCR");
  ses2 <- SES(target, dataset, max_k = 1, threshold = 0.05, test = "censIndWR");
  ses3 <- SES(target, dataset, max_k = 1, threshold = 0.05, test = "censIndER");
```

`Constraint based feature selection algorithms`
SES: Feature selection algorithm for identifying multiple minimal, statistically-equivalent and equally-predictive feature signatures MMPC: Feature selection algorithm for identifying minimal feature subsets

---

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. MMPC algorithm follows the same approach without generating multiple feature subsets.

Usage

```
SES(target, dataset, max_k = 3, threshold = 0.05, test = NULL, ini = NULL, wei = NULL,
 user_test = NULL, hash = FALSE, hashObject = NULL, robust = FALSE, ncores = 1)

MMPC(target, dataset, max_k = 3, threshold = 0.05, test = NULL, ini = NULL, wei = NULL,
 user_test = NULL, hash = FALSE, hashObject = NULL, robust = FALSE, ncores = 1,
 backward = FALSE)
```

Arguments

`target`	The class variable. Provide either a string, an integer, a numeric 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. See also `CondIndTests`.
`ini`	This is a supposed to be a list. After running SES or MMPC with some hyper-parameters you might want to run SES again with different hyper-parameters. To avoid calculating the univariate associations (first step of SES and of MPPC) again, you can extract them from the first run of SES and plug them here. This can speed up the second run (and subequent runs of course) by 50%. See the details and the argument "univ" in the output values.
`wei`	A vector of weights to be used for weighted regression. The default value is NULL. It is not suggested when robust is set to TRUE.

`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 or MMPC. 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 or MMPC. Important: the generated hashObjects should be used only when the same dataset is re-analyzed, possibly with different values of max_k and threshold.
`robust`	A boolean variable which indicates whether (TRUE) or not (FALSE) to use a robust version of the statistical test if it is available. It takes more time than a non robust version but it is suggested in case of outliers. Default value is FALSE.
`ncores`	How many cores to use. This plays an important role if you have tens of thousands of variables or really large sample sizes and tens of thousands of variables and a regression based test which requires numerical optimisation. In other cases it will not make a difference in the overall time (in fact it can be slower). The parallel computation is used in the first step of the algorithm, where univariate associations are examined, those take place in parallel. We have seen a reduction in time of 50% with 4 cores in comparison to 1 core. Note also, that the amount of reduction is not linear in the number of cores.
`backward`	If TRUE, the backward (or symmetry correction) phase will be implemented. This removes any falsely included variables in the parents and children set of the target variable and it will slow down the algorithm. Bear in mind that the target becomes predictor variables. Hence, this is advised to be used with "testIndifisher" and "gSquare" only and not with survival or multivariate targets. This is because the two aforementioned tests are symmetrical, i.e. there is not dependent or independent variable. In addition, if there are highly collinear (or statistically equivalent) variables, this phase tends to remove correctly identified variables, simply because it will identify a variable wich is highly collinear with the target variable.

Details

The SES 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/

The MMPC function mplements the MMPC algorithm as presented in "Tsamardinos, Brown and Aliferis. The max-min hill-climbing Bayesian network structure learning algorithm" http://www.dsl-lab.org/supplements/mmhc_paper/paper_online.pdf

For faster computations in the internal SES functions, install the suggested package **gRbase**. In addition, the output value "univ" along with the output value "hashObject" can speed up the computations of subesequent runs of SES and MMPC. The first run with a specific pair of hyper-parameters (threshold and max_k) the univariate associations tests and the conditional independence tests (test statistic and logarithm of their corresponding p-values) are stored and returned. In the next run(s) with different pair(s) of hyper-parameters you can use this information to save time. With a few thousands of variables you will see the difference, which can be up to 50%. For the non robust correlation based tests, the difference may not be significant though, because a Fortran code is used to extract the (unconditional) correlation coefficients.

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 observations) the max_k parameter should be ≤ 5, otherwise the conditional independence test may not be able to provide reliable results.

If the dataset (predictor variables) 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 the target is a factor, the multinomial or the binary logistic regression is used. If the target has two values only, binary logistic regression will be used.
- if target is a ordered factor, ordinal regression is used in the logistic test. Hence, if you want to use multinomial or ordinal logistic regression, make sure your target is factor.
- if target is a numerical vector and the dataset is a matrix or a data.frame with continuous variables, the Fisher conditional independence test is used. If the dataset is a data.frame and there are categorical variables, linear regression is used.
- if target is discrete numerical (counts), the poisson regression conditional independence test is used. If there are only two values, the binary logistic regression is to be used.
- if target is a Surv object, a Survival conditional independence test is used.
- if target is a matrix with at least 2 columns, the multivariate linear regression is used.
- if target is a 2 column matrix whose columns are the number of successes and the number of trials (first and second column respectively) the testIndBinom should be 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.

For all the available conditional independence tests that are currently included on the package, please see `CondIndTests`.

If two or more p-values are below the machine epsilon (.Machine$double.eps which is equal to 2.220446e-16), all of them are set to 0. To make the comparison or the ordering feasible we use the logarithm of the p-value. The max-min heuristic though, requires comparison and an ordering of the p-values. Hence, all conditional independence tests calculate the logarithm of the p-value.

If there are missing values in the dataset (predictor variables) columnwise imputation takes place. The median is used for the continuous variables and the mode for categorical variables. It is a naive and not so clever method. For this reason the user is encouraged to make sure his data contain no missing values.

If you have percentages, in the (0, 1) interval, they are automatically mapped into R by using the logit transformation. If you set the test to `testIndBeta`, beta regression is used. If you have compositional data, positive multivariate data where each vector sums to 1, with NO zeros, they are also mapped into the Euclidean space using the additive log-ratio (multivariate logit) transformation (Aitchison, 1986).

If you use testIndSpearman (argument "test"), the ranks of the data calculated and those are used in the caclulations. This speeds up the whole procedure.

As a rule of thumb you can try this. If for example you have counts and want to see which model fits best, there are two ways. Calculate the mean and the variance. If they are similar, use the Poisson instead of the negative binomial as it is much faster. If you are not convinced, you can either use the negative binomial or do the following simulation study.

x <- matrix(rnorm(n * 1000), ncol = 1000) a <- Rfast::univglms(y, x) hist(a[, 2]) ## histogram of the p-values

If the histogram shows a uniform distribution, use the Poisson regression. If the histogram is not uniform, then repeat the simluation but with a negative binomial distribution. If the histogram is again not flat, then another model is necessary. If the data come from a Poisson or negative binomial, the histogram with a negative binomial regressino will be flat. If the data come a from a negative binomial, the histogram with a Poisson will not be uniform.

On the same page, if you have many zeros, try Rfast::zip.mle and see whether there are grounds to to facilitate the use of a zero inflated Poisson model. Otherwise, do a simulation study like before.

Value

The output of the algorithm is an object of the class 'SESoutput' for SES or 'MMPCoutput' for MMPC 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. Featured only in SES.
`signatures`	A matrix reporting all equivalent signatures (one signature for each row). Featured only in SES.
`hashObject`	The hashObject caching the statistic calculated 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).
`univ`	This is a list with the univariate associations. The test statistics and their corresponding logged p-values, along with their flag (1 if the test was perfromed and 0 otherwise). This list is very important for subsequent runs of SES with different hyper-parameters. After running SES with some hyper-parameters you might want to run SES again with different hyper-parameters. To avoid calculating the univariate associations (first step of SES or MMPC) again, you can take

	this list from the first run of SES and plug it in the argument "ini" in the next run(s) of SES or MMPC. This can speed up the second run (and subequent runs of course) by 50%. See the argument "univ" in the output values.
`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 second element is the system time and the third element is the elapsed time.
`test`	The character name of the statistic test used.
`rob`	The value of the robust option, either TRUE or FALSE.

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 by the SES and MMPC algorithm operations are:

hash : for the hash-based implementation

quantreg: for the quantile (median) regression

betareg: for beta regression

MASS: for negative binomial regression

pscl: for zero inlfated poisson regression

nnet : also require(stats) and require(MASS) for the testIndLogistic test

and **ordinal** : also require(stats) and require(MASS) for the testIndLogistic test

survival : for the censIndCR, censIndWR and the censIndER tests

doParallel: for parallel computations

Author(s)

Ioannis Tsamardinos, Vincenzo Lagani

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

Tsamardinos, Brown and Aliferis (2006). The max-min hill-climbing Bayesian network structure learning algorithm. Machine learning, 65(1), 31-78.

See Also

CondIndTests, cv.ses

Examples

```
set.seed(123)
require("hash", quietly = TRUE)

#simulate a dataset with continuous data
dataset <- matrix(runif(1000 * 200, 1, 100), ncol = 200)

#define a simulated class variable
target <- 3 * dataset[, 10] + 2 * dataset[, 100] + 3 * dataset[, 20] + rnorm(1000, 0, 5)

# define some simulated equivalences
dataset[, 15] <- dataset[, 10] + rnorm(1000, 0, 2)
dataset[, 10] <- dataset[ , 10] + rnorm(1000, 0, 2)
dataset[, 150] <- dataset[, 100] + rnorm(1000, 0, 2)
dataset[, 130] <- dataset[, 100] + rnorm(1000, 0, 2)

# run the SES algorithm
sesObject <- SES(target , dataset, max_k = 5, threshold = 0.05, 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;

# 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);

sesObject3 <- SES(target, dataset, max_k = 2, threshold = 0.01, test = "testIndFisher",
ini = sesObject@univ, hash = TRUE, hashObject = hashObj);
```

```
# retrieve the results: summary, plot, sesObject2@...)
summary(sesObject2)
# get the run time
sesObject@runtime;
sesObject2@runtime;
sesObject3@runtime;


# MMPC algorithm
mmpcObject <- MMPC(target, dataset, max_k = 3, threshold = 0.05, test="testIndFisher");
mmpcObject@selectedVars
mmpcObject@runtime
```

`Constraint based feature selection algorithms for longitudinal and clustered data`
SES.temporal: Feature selection algorithm for identifying multiple minimal, statistically-equivalent and equally-predictive feature signatures MMPC.temporal: Feature selection algorithm for identifying minimal feature subsets

---

Description

SES.temporal 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. MMPC.temporal algorithm follows the same approach without generating multiple feature subsets. They are both adapted to longitudinal target variables.

Usage

```
SES.temporal(target, reps = NULL, group, dataset, max_k = 3, threshold = 0.05,
test = NULL, ini = NULL, wei = NULL, user_test = NULL, hash = FALSE, hashObject = NULL,
slopes = FALSE, ncores = 1)

MMPC.temporal(target, reps = NULL, group, dataset, max_k = 3, threshold = 0.05,
test = NULL, ini = NULL, wei = NULL, user_test = NULL, hash = FALSE, hashObject = NULL,
slopes = FALSE, ncores = 1)
```

Arguments

`target`	The class variable. Provide a vector with continuous (normal), binary (binomial) or discrete (Poisson) data.
`reps`	A numeric vector containing the time points of the subjects. It's length is equal to the length of the target variable. If you have clustered data, leave this NULL.
`group`	A numeric vector containing the subjects or groups. It must be of the same legnth as target.

`dataset`	The data-set; provide either a data frame or a matrix (columns = variables , rows = samples). Currently, only continuous datasets are supported. 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. Currently, the only available conditional independence test is the `testIndGLMM`, which fits linear mixed models.
`ini`	This is a supposed to be a list. After running SES or MMPC with some hyper-parameters you might want to run SES again with different hyper-parameters. To avoid calculating the univariate associations (first step of SES and of MPPC) again, you can extract them from the first run of SES and plug them here. This can speed up the second run (and subequent runs of course) by 50%. See the details and the argument "univ" in the output values.
`wei`	A vector of weights to be used for weighted regression. The default value is NULL.
`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.temporal. 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.
`slopes`	Should random slopes for the ime effect be fitted as well? Default value is FALSE.
`ncores`	How many cores to use. This plays an important role if you have tens of thousands of variables or really large sample sizes and tens of thousands of variables and a regression based test which requires numerical optimisation. In other cases it will not make a difference in the overall time (in fact it can be slower). The parallel computation is used in the first step of the algorithm, where univariate associations are examined, those take place in parallel. We have seen a reduction in time of 50% with 4 cores in comparison to 1 core. Note also, that the amount of reduction is definetely not linear in the number of cores.

Details

The SES.temporal function implements the Statistically Equivalent Signature (SES) algorithm as presented in "Tsamardinos, Lagani and Pappas, HSCBB 2012" adapted to longitudinal data.

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

The MMPC function mplements the MMPC algorithm as presented in "Tsamardinos, Brown and Aliferis. The max-min hill-climbing Bayesian network structure learning algorithm" adapted to longitudinal data. http://www.dsl-lab.org/supplements/mmhc_paper/paper_online.pdf

For faster computations in the internal SES functions, install the suggested package "**gRbase**". In addition, the output value "univ" along with the output value "hashObject" can speed up the computations of subesequent runs of SES and MMPC. The first run with a specific pair of hyper-parameters (threshold and max_k) the univariate associations tests and the conditional independence tests (test statistic and logarithm of their corresponding p-values) are stored and returned. In the next run(s) with different pair(s) of hyper-parameters you can use this information to save time. With a few thousands of variables you will see the difference, which can be up to 50%.

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 observations) 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 only available, which is `testIndGLMM`.

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.

For all the available conditional independence tests that are currently included on the package, please see "?CondIndTests".

If two or more p-values are below the machine epsilon (.Machine$double.eps which is equal to 2.220446e-16), all of them are set to 0. To make the comparison or the ordering feasible we use the logarithm of the p-value. The max-min heuristic though, requires comparison and an ordering of the p-values. Hence, all conditional independence tests calculate the logarithm of the p-value.

If there are missing values in the dataset (predictor variables) columnwise imputation takes place. The median is used for the continuous variables and the mode for categorical variables. It is a naive and not so clever method. For this reason the user is encouraged to make sure his data contain no missing values.

If you have percentages, in the (0, 1) interval, they are automatically mapped into R by using the logit transformation and a linear mixed model is fitted. If you have binary data, logistic mixed regression is applied and if you have discrete data (counts), Poisson mixed regression is applied.

Value

The output of the algorithm is an object of the class 'SES.temporal.output' for SES.temporal or 'MMPC.temporal.output' for MMPC.temporal 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. Featured only in SES.
`signatures`	A matrix reporting all equivalent signatures (one signature for each row). Featured only in SES.
`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).
`univ`	This is a list with the univariate associations. The test statistics and their corresponding logge p-values, along with their flag (1 if the test was perfromed and 0 otherwise). This list is very important for subsequent runs of SES with different hyper-parameters. After running SES with some hyper-parameters you might want to run SES again with different hyper-parameters. To avoid calculating the univariate associations (first step of SES or MMPC) again, you can take this list from the first run of SES and plug it in the argument "ini" in the next run(s) of SES or MMPC. This can speed up the second run (and subequent runs of course) by 50%. See the argument "univ" in the output values.
`max_k`	The max_k option used in the current run.
`threshold`	The threshold option used in the current run.
`slope`	Whether random slopes for the time effects were used or not, TRUE or FALSE.
`runtime`	The run time of the algorithm. A numeric vector. The first element is the user time, the second element is the system time and the third element is the elapsed time.

Generic Functions implemented for SESoutput Object:

`summary(x=SES.temporal.output)`
: Summary view of the SES.temporal.output object.

`plot(object=SES.temporal.output, 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 only package required for the SES.temporal and MMPC.temporal algorithm operations is the **lme4** and the **doParallel** for parallel computations.

Author(s)

Ioannis Tsamardinos, Vincenzo Lagani

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

Tsamardinos, Brown and Aliferis (2006). The max-min hill-climbing Bayesian network structure learning algorithm. Machine learning, 65(1), 31-78.

I. Tsamardinos, M. Tsagris and V. Lagani (2015). Feature selection for longitudinal data. Proceedings of the 10th conference of the Hellenic Society for Computational Biology & Bioinformatics (HSCBB15)

Pinheiro J. and D. Bates. Mixed-effects models in S and S-PLUS. Springer Science \& Business Media, 2006.

See Also

`CondIndTests`, `testIndGLMM`

Examples

```
## require(lme4)
## data(sleepstudy)
## attach(sleepstudy)
## x <- matrix(rnorm(180 * 100),ncol = 100) ## unrelated preidctor variables
## m1 <- SES.temporal(Reaction, Days, Subject, x)
## m2 <- MMPC.temporal(Reaction, Days, Subject, x)
```

Constraint based feature selection algorithms for multiple datasets

ma.ses: Feature selection algorithm for identifying multiple minimal, statistically-equivalent and equally-predictive feature signatures with multiple datasets ma.mmpc: Feature selection algorithm for identifying minimal feature subsets with multiple datasets

---

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 two or more high dimensional datasets. See also Details. MMPC algorithm follows the same approach without generating multiple feature subsets.

Usage

```
ma.ses(target, dataset, ina, statistic = FALSE, max_k = 3, threshold = 0.05,
test = NULL, ini = NULL, user_test = NULL, hash = FALSE, hashObject = NULL,
robust = FALSE, ncores = 1)

ma.mmpc(target, dataset, ina, statistic = FALSE, max_k = 3, threshold = 0.05,
test = NULL, ini = NULL, user_test = NULL, hash = FALSE, hashObject = NULL,
robust = FALSE, ncores = 1, backward = FALSE)
```

Arguments

`target`	The class variable. Provide either a string, an integer, a numeric 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).
`ina`	A numerical vector indicating the dataset. The numbers must be 1, 2, 3,...
`statistic`	A boolean variable indicating whether the test statistics (TRUE) or the p-values should be combined (FALSE). See the details about this.
`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. See also `CondIndTests`.
`ini`	This is a supposed to be a list. After running SES or MMPC with some hyper-parameters you might want to run SES again with different hyper-parameters. To avoid calculating the univariate associations (first step of SES and of MPPC) again, you can extract them from the first run of SES and plug them here. This can speed up the second run (and subequent runs of course) by 50%. See the details and the argument "univ" in the output values.
`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 or MMPC. 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 or MMPC. Important: the generated hashObjects should be used only when the same dataset is re-analyzed, possibly with different values of max_k and threshold.
`robust`	A boolean variable which indicates whether (TRUE) or not (FALSE) to use a robust version of the statistical test if it is available. It takes more time than a non robust version but it is suggested in case of outliers. Default value is FALSE.
`ncores`	How many cores to use. This plays an important role if you have tens of thousands of variables or really large sample sizes and tens of thousands of variables and a regression based test which requires numerical optimisation. In other cases it will not make a difference in the overall time (in fact it can be slower). The parallel computation is used in the first step of the algorithm, where univariate associations are examined, those take place in parallel. We have seen a reduction in time of 50% with 4 cores in comparison to 1 core. Note also, that the amount of reduction is not linear in the number of cores.
`backward`	If TRUE, the backward (or symmetry correction) phase will be implemented. This removes any falsely included variables in the parents and children set of the

target variable and it will slow down the algorithm. Bear in mind that the target becomes predictor variables. Hence, this is advised to be used with "testIndi-fisher" and "gSquare" only and not with survival or multivariate targets. This is because the two aforementioned tests are symmetrical, i.e. there is not dependent or independent variable. In addition, if there are highly collinear (or statistically equivalent) variables, this phase tends to remove correctly identified variables, simply because it will identify a variable wich is highly collinear with the target variable.

Details

This is more at an experimental stage at the present.

The SES 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/

The MMPC function mplements the MMPC algorithm as presented in "Tsamardinos, Brown and Aliferis. The max-min hill-climbing Bayesian network structure learning algorithm" http://www.dsl-lab.org/supplements/mmhc_paper/paper_online.pdf

For faster computations in the internal SES functions, install the suggested package "**gRbase**". In addition, the output value "univ" along with the output value "hashObject" can speed up the computations of subesequent runs of SES and MMPC. The first run with a specific pair of hyper-parameters (threshold and max_k) the univariate associations tests and the conditional independence tests (test statistic and logarithm of their corresponding p-values) are stored and returned. In the next run(s) with different pair(s) of hyper-parameters you can use this information to save time. With a few thousands of variables you will see the difference, which can be up to 50%. For the non robust correlation based tests, the difference may not be significant though, because a Fortran code is used to extract the (unconditional) correlation coefficients.

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 observations) the max_k parameter should be ≤ 5, otherwise the conditional independence test may not be able to provide reliable results.

If the dataset (predictor variables) 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.

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.

For all the available conditional independence tests that are currently included on the package, please see `CondIndTests`.

If two or more p-values are below the machine epsilon (.Machine$double.eps which is equal to 2.220446e-16), all of them are set to 0. To make the comparison or the ordering feasible we use the logarithm of the p-value. The max-min heuristic though, requires comparison and an ordering of the p-values. Hence, all conditional independence tests calculate the logarithm of the p-value.

If there are missing values in the dataset (predictor variables) columnwise imputation takes place. The median is used for the continuous variables and the mode for categorical variables. It is a naive and not so clever method. For this reason the user is encouraged to make sure his data contain no missing values.

If you have percentages, in the (0, 1) interval, they are automatically mapped into R by using the logit transformation.

If you use testIndSpearman (argument "test"), the ranks of the data calculated and those are used in the caclulations. This speeds up the whole procedure.

Currently only the testIndFisher and testIndSpearman tests are supported for use in the algorithm.

If the argument **statistic** is set to FALSE, the p-values from the hypothesis test of each dataset are combined via Fisher's meta-analytic approach, that is $T = -2\sum_{i=1}^{k} \log p_i$ and $T\ \chi^2_{2k}$. If **statistic** is TRUE, the test statistics are combined as $T = \frac{\sum_{i=1}^{k} t_i/se(t_i)}{\sum_{i=1}^{k} 1/se(t_i)}$ and $T\ N(0,1)$.

Value

The output of the algorithm is an object of the class 'SESoutput' for SES or 'MMPCoutput' for MMPC 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. Featured only in SES.
`signatures`	A matrix reporting all equivalent signatures (one signature for each row). Featured only in SES.
`hashObject`	The hashObject caching the statistic calculated 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).
`univ`	This is a list with the univariate associations. The test statistics and their corresponding logged p-values, along with their flag (1 if the test was perfromed and 0 otherwise). This list is very important for subsequent runs of SES with different hyper-parameters. After running SES with some hyper-parameters you might want to run SES again with different hyper-parameters. To avoid calculating the univariate associations (first step of SES or MMPC) again, you can take this list from the first run of SES and plug it in the argument "ini" in the next run(s) of SES or MMPC. This can speed up the second run (and subequent runs of course) by 50%. See the argument "univ" in the output values.
`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 second element is the system time and the third element is the elapsed time.

`test`	The character name of the statistic test used.
`rob`	The value of the robust option, either TRUE or FALSE.

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.

Author(s)

Ioannis Tsamardinos, Vincenzo Lagani

R implementation and documentation: Giorgos Athineou <athineou@csd.uoc.gr>, Vincenzo Lagani <vlagani@csd.uoc.gr> and Michail Tsagris <mtsagris@csd.uoc.gr>

References

V. Lagani, A.D. Karozou, D. Gomez-Cabrero, G. Silberberg and I. Tsamardinos (2016). A comparative evaluation of data-merging and meta-analysis methods for reconstructing gene-gene interactions, BMC Bioinformatics 17(Supplementary 5): 287-305.

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.

Tsamardinos, Brown and Aliferis (2006). The max-min hill-climbing Bayesian network structure learning algorithm. Machine learning, 65(1), 31-78.

See Also

SES, CondIndTests, cv.ses

Examples

```
set.seed(123)
require(hash)

#simulate a dataset with continuous data
dataset <- matrix(runif(1000 * 100, 1, 100), ncol = 100)

#define a simulated class variable
target <- 3 * dataset[, 10] + 2 * dataset[, 20] + 3 * dataset[, 30] + rnorm(1000, 0, 5)

#define some simulated equivalences
dataset[, 15] <- dataset[, 10] + rnorm(1000, 0, 2)
dataset[, 10] <- dataset[ , 10] + rnorm(1000, 0, 2)
dataset[, 25] <- dataset[, 20] + rnorm(1000, 0, 2)
dataset[, 23] <- dataset[, 20] + rnorm(1000, 0, 2)
```

```
require("hash", quietly = TRUE)
#run the SES algorithm
a1 <- SES(target , dataset, max_k = 5, threshold = 0.05, test = "testIndFisher",
hash = TRUE, hashObject = NULL)

ina <- rbinom(1000, 2, 0.5) + 1
a2 <- ma.ses(target , dataset, ina = ina, max_k = 5, threshold = 0.05, test = "testIndFisher",
hash = TRUE, hashObject = NULL)
a3 <- ma.mmpc(target , dataset, ina = ina, max_k = 5, threshold = 0.05, test = "testIndFisher",
hash = TRUE, hashObject = NULL)

#get the generated signatures
a1@signatures
a2@signatures
a3@selectedVars
```

Correlation based conditonal independence tests

Fisher and Spearman conditional independence test for continuous 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, wei = NULL, statistic = FALSE,
dataInfo = NULL, univariateModels = NULL, hash = FALSE, stat_hash = NULL,
pvalue_hash = NULL, robust = FALSE)

testIndSpearman(target, dataset, xIndex, csIndex, wei = NULL, statistic = FALSE,
dataInfo = NULL, univariateModels = NULL, hash = FALSE, stat_hash = NULL,
pvalue_hash = NULL, robust = FALSE)

permFisher(target, dataset, xIndex, csIndex, wei = NULL, statistic = FALSE,
dataInfo = NULL, univariateModels = NULL, hash = FALSE, stat_hash = NULL,
pvalue_hash = NULL, robust = FALSE, R = 999)
```

Arguments

`target`	A numeric vector containing the values of the target variable. If the values are proportions or percentages, i.e. strictly within 0 and 1 they are mapped into R using log(target/(1 - target)). This can also be a list of vectors as well. In this case, the metanalytic approach is used.

`dataset`	A numeric 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.
`wei`	A vector of weights to be used for weighted regression. The default value is NULL.
`statistic`	A boolean variable indicating whether the test statistics (TRUE) or the p-values should be combined (FALSE). See the details about this. For the permFisher test this is not taken into account.
`dataInfo`	A 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.
`robust`	A boolean variable which indicates whether (TRUE) or not (FALSE) to use a robustified version of Fisher's correlation coefficient via MM-estimation available from `rlm` in the package "MASS". Two regressions are fitted and the square root ot the absolute value of the beta coefficients is used to calculate the correlation coefficient (Shevlyakov and Smirnov, 2011). For the conditional correlation the correlation of the residuals of the two robust regressions is calcualted. For more ways of calculating the correlation coefficient see the references. It takes more time than non robust version but it is suggested in case of outliers. Default value is FALSE. In the case of testIndSpearman, this is not used, as Spearman correlation is robust by default.
`R`	The number of permutations to use. The default value is 999.

Details

If hash = TRUE, testIndFisher 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.

For all the available conditional independence tests that are currently included on the package, please see "?CondIndTests".

Note that if the `testIndReg` is used instead the results will not be be the same, unless the sample size is very large. This is because the Fisher test uses the t distribution stemming from the Fisher's z transform and not the t distribution of the correlation coefficient.

BE CAREFUL with testIndSpearman. The Pearson's correlation coefficient is actually calculated. So, you must have transformed the data into their ranks before plugging them here. The reason for this is to speed up the computation time, as this test can be used in SES, MMPC and mmhc.skel. The variance of the Fisher transformed Spearman's correlation is $\frac{1.06}{n-3}$ and the variance of the Fisher transformed Pearson's correlation coefficient is $\frac{1}{n-3}$.

When performing the above tests with multiple datasets, the test statistic and the p-values are combined in a meta-analytic way. Is up to the user to decide whether to use the fixed effects model approach and combine the test statistics (statistic = TRUE), or combine the p-values as Fisher suggested (statistic = FALSE).

The argument R is useful only for the permFisher test.

Value

A list including:

`pvalue`	A numeric value that represents the logarithm of 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

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

References

Fisher R. A. (1925). Statistical methods for research workers. Genesis Publishing Pvt Ltd.

Fisher R. A. (1948). Combining independent tests of significance. American Statistician, 2(5), 30–31

Fisher R. A. (1915). Frequency distribution of the values of the correlation coefficient in samples from an indefinitely large population. Biometrika, 10(4): 507–521.

Fieller E. C., Hartley H. O. and Pearson E. S. (1957). Tests for rank correlation coefficients. I. Biometrika, 44(3/4): 470–481.

Fieller E. C. and Pearson E. S. (1961). Tests for rank correlation coefficients. II. Biometrika, 48(1/2): 29–40.

Hampel F. R., Ronchetti E. M., Rousseeuw P. J., and Stahel W. A. (1986). Robust statistics: the approach based on influence functions. John Wiley & Sons.

Pearson, K. (1895). Note on regression and inheritance in the case of two parents. Proceedings of the Royal Society of London, 58, 240–242.

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

Lee Rodgers J., and Nicewander W.A. (1988). "Thirteen ways to look at the correlation coefficient." The American Statistician 42(1): 59–66.

Shevlyakov G. and Smirnov P. (2011). Robust Estimation of the Correlation Coefficient: An Attempt of Survey. Austrian Journal of Statistics, 40(1 & 2): 147–156.

See Also

`testIndSpearman`, `testIndReg`, `SES`, `testIndLogistic`, `gSquare`, `CondIndTests`

Examples

```
#simulate a dataset with continuous data
dataset <- matrix(runif(1000 * 200, 1, 1000), nrow = 1000 )
#the target feature is the last column of the dataset as a vector
target <- dataset[, 200]
res1 <- testIndFisher(target, dataset, xIndex = 44, csIndex = 100)
res2 <- testIndSpearman(target, dataset, xIndex = 44, csIndex = 100)
res3 <- permFisher(target, dataset, xIndex = 44, csIndex = 100, R = 999)


#define class variable (here tha last column of the dataset)
dataset <- dataset[, -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");
```

`Correlation based tests with and without permutation p-value`

Conditional independence test for continuous class variables with and without permutation based p-value

---

Description

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

Usage

```
condi(ind1, ind2, cs, dat, type = "pearson", rob = FALSE, R = 1)
dist.condi(ind1, ind2, cs, dat, type = NULL, rob = NULL, R = 499)
```

Arguments

`ind1`	The index of the one variable to be considered.
`ind2`	The index of the other variable to be considered.
`cs`	The index or indices of the conditioning set of variable(s).
`dat`	A matrix with the data.
`type`	Do you want the Pearson (type = "pearson") or the Spearman (type = "spearman") correlation to be used. For "dist.condi" this is an obsolete argument but it requires to exist when it is used in the PC algorithm.
`rob`	If you choose type="pearson" then you can sapecify whether you want a robust version of it (TRUE) or not (FALSE). For "dist.condi" this is an obsolete argument but it requires to exist when it is used in the PC algorithm.
`R`	If R = 1 then the asymptotic p-value is calculated. If R > 1 a permutation based p-value is returned. For the distance correlation based test, this is set to 499 by default and is used in the partial correlaiton test only.

Details

This test is currently designed for usage by the PC algorithm. The Fisher conditional independence test which is based on the Pearson or Spearman correlation coefficients is much faster than the distance based (partial) correlation test.

The distance correlation can handle non linear relationships as well. The p-value for the partial distance correlation is calculated via permutations and is slow.

Value

A vector including the test statistic, it's associated p-value and the relevant degrees of freedom. In the case of a permutation based p-value, the returned test statistic is the observed test statistic divided by the relevant degrees of freedom (Pearson and Spearman correlation coefficients only). This is for the case of ties between many permutation based p-values. The PC algorithm choose a pair of variables based on the p-values. If they are equal it will use the test statistic.

Author(s)

Michail Tsagris

R implementation and documentation: Giorgos Athineou <athineou@csd.uoc.gr> and Michail Tsagris <mtsagris@csd.uoc.gr>

References

Hampel F. R., Ronchetti E. M., Rousseeuw P. J., and Stahel W. A. (1986). Robust statistics: the approach based on influence functions. John Wiley & Sons.

Lee Rodgers J., and Nicewander W.A. (1988). "Thirteen ways to look at the correlation coefficient". The American Statistician 42(1): 59-66.

Shevlyakov G. and Smirnov P. (2011). Robust Estimation of the Correlation Coefficient: An Attempt of Survey. Austrian Journal of Statistics, 40(1 & 2): 147-156.

Spirtes P., Glymour C. and Scheines R. Causation, Prediction, and Search. The MIT Press, Cambridge, MA, USA, second edition, January 2001.

Szekely G.J. and Rizzo, M.L. (2014). Partial distance correlation with methods for dissimilarities. The Annals of Statistics, 42(6): 2382–2412.

Szekely G.J. and Rizzo M.L. (2013). Energy statistics: A class of statistics based on distances. Journal of Statistical Planning and Inference 143(8): 1249–1272.

See Also

`testIndFisher`, `testIndSpearman`, `pc.skel`, `gSquare`, `CondIndTests`

Examples

```
#simulate a dataset with continuous data
dataset <- matrix(runif(500 * 5, 1, 100), nrow = 5 )
testIndFisher(dataset[, 1], dataset[, -1], xIndex = 1, csIndex = 2)
condi(ind1 = 1, ind2 = 2, cs = 3, dataset, R = 1)
condi(ind1 = 1, ind2 = 2, cs = 3, dataset, R = 999)
dist.condi(ind1 = 1, ind2 = 2, 0, dataset)
dist.condi(ind1 = 1, ind2 = 2, cs = 3, dataset, R = 99)
```

`Cross-validation for ridge regression`
Cross validation for the ridge regression

Description

Cross validation for the ridge regression is performed using the TT estimate of bias (Tibshirani and Tibshirani, 2009). There is an option for the GCV criterion which is automatic.

Usage

```
ridgereg.cv( target, dataset, K = 10, lambda = seq(0, 2, by = 0.1), auto = FALSE,
seed = FALSE, ncores = 1, mat = NULL )
```

Arguments

`target`	A numeric vector containing the values of the target variable. If the values are proportions or percentages, i.e. strictly within 0 and 1 they are mapped into R using log(target/(1 - target)).
`dataset`	A numeric matrix containing the variables. Rows are samples and columns are features.
`K`	The number of folds. Set to 10 by default.
`lambda`	A vector with the a grid of values of λ to be used.
`auto`	A boolean variable. If it is TRUE the GCV criterion will provide an automatic answer for the best $lambda$. Otherwise k-fold cross validation is performed.
`seed`	A boolean variable. If it is TRUE the results will always be the same.
`ncores`	The number of cores to use. If it is more than 1 parallel computing is performed.
`mat`	If the user has its own matrix with the folds, he can put it here. It must be a matrix with K columns, each column is a fold and it contains the positions of the data, i.e. numbers, not the data. For example the first column is c(1,10,4,25,30), the second is c(21, 23,2, 19, 9) and so on.

Details

The lm.ridge command in MASS library is a wrapper for this function. If you want a fast choice of λ, then specify auto = TRUE and the λ which minimizes the generalised cross-validation criterion will be returned. Otherise a k-fold cross validation is performed and the estimated performance is bias corrected as suggested by Tibshirani and Tibshirani (2009).

Value

A list including:

`mspe`	If auto is FALSE the values of the mean prediction error for each value of λ.
`lambda`	If auto is FALSE the λ which minimizes the MSPE.
`performance`	If auto is FALSE the minimum bias corrected MSPE along with the estimate of bias.
`runtime`	The run time of the algorithm. A numeric vector. The first element is the user time, the second element is the system time and the third element is the elapsed time.

Note

The values can be extracted with the $ symbol, i.e. this is an S3 class output.

Author(s)

Michail Tsagris

R implementation and documentation: Michail Tsagris <mtsagris@csd.uoc.gr>

References

Hoerl A.E. and R.W. Kennard (1970). Ridge regression: Biased estimation for nonorthogonal problems. Technometrics, 12(1):55-67.

Brown P. J. (1994). Measurement, Regression and Calibration. Oxford Science Publications.

Tibshirani R.J., and Tibshirani R. (2009). A bias correction for the minimum error rate in cross-validation. The Annals of Applied Statistics 3(2): 822-829.

See Also

ridge.reg

Examples

```
#simulate a dataset with continuous data
dataset <- matrix(runif(200 * 50, 1, 100), nrow = 200 )
#the target feature is the last column of the dataset as a vector
target <- dataset[, 50]
a1 <- ridgereg.cv(target, dataset, auto = TRUE)
a2 <- ridgereg.cv( target, dataset, K = 10, lambda = seq(0, 1, by = 0.1) )
```

Cross-Validation for SES and MMPC

Cross-Validation for SES and MMPC

Description

The function performs a k-fold cross-validation for identifying the best values for the SES and MMPC 'max_k' and 'threshold' hyper-parameters.

Usage

```
cv.ses(target, dataset, wei = NULL, kfolds = 10, folds = NULL,
alphas = c(0.1, 0.05, 0.01), max_ks = c(3, 2), task = NULL,
metric = NULL, modeler = NULL, ses_test = NULL, ncores = 1)

cv.mmpc(target, dataset, wei = NULL, kfolds = 10, folds = NULL,
alphas = c(0.1, 0.05, 0.01), max_ks = c(3, 2), task = NULL,
metric = NULL, modeler = NULL, mmpc_test = NULL, ncores = 1)
```

Arguments

`target`	The target or class variable as in SES and MMPC.
`dataset`	The dataset object as in SES and MMPC.
`wei`	A vector of weights to be used for weighted regression. The default value is NULL.

`kfolds`	The number of the folds in the k-fold Cross Validation (integer).
`folds`	The folds of the data to use (a list generated by the function generateCVRuns TunePareto). If NULL the folds are created internally with the same function.
`alphas`	A vector of SES or MMPC thresholds hyper parameters used in CV.
`max_ks`	A vector of SES or MMPC max_ks parameters used in CV.
`task`	A character ("C", "R" or "S"). It can be "C" for classification (logistic, multinomial or ordinal regression), "R" for regression (robust and non robust linear regression, median regression, (zero inflated) poisson and negative binomial regression, beta regression), "S" for survival regresion (Cox, Weibull or exponential regression).
`metric`	A metric function provided by the user. If NULL the following functions will be used: auc.mxm, mse.mxm, ci.mxm for classification, regression and survival analysis tasks, respectively. See details for more. If you know what you have put it here to avoid the function chopsing somehting else. **Note** that you put these words as they are, without "".
`modeler`	A modeling function provided by the user. If NULL the following functions will be used: glm.mxm, lm.mxm, coxph.mxm for classification, regression and survival analysis tasks, respectively. See details for more. If you know what you have put it here to avoid the function chopsing somehting else. **Note** that you put these words as they are, without "".
`ses_test`	A function object that defines the conditional independence test used in the SES function (see also SES help page). If NULL, testIndFisher, testIndLogistic and censIndLR are used for classification, regression and survival analysis tasks, respectively. If you know what you have put it here to avoid the function chopsing somehting else.
`mmpc_test`	A function object that defines the conditional independence test used in the MMPC function (see also SES help page). If NULL, testIndFisher, testIndLogistic and censIndLR are used for classification, regression and survival analysis tasks, respectively.
`ncores`	This argument is valid only if you have a multi-threaded machine.

Details

Input for metric functions: predictions: A vector of predictions to be tested. test_target: target variable actual values to be compared with the predictions.

The output of a metric function is a single numeric value. **Higher values indicate better performance**. Metric based on error measures should be modified accordingly (e.g., multiplying the error for -1)

The metric functions that are currently supported are:

- auc.mxm: "area under the receiver operator characteristic curve" metric, as provided in the package ROCR.
- acc.mxm: accuracy metric.
- mse.mxm: -1 * (mean squared error), for robust and non robust linear regression and median (quantile) regression.

- ci.mxm: 1 - concordance index as provided in the rcorr.cens function from the Hmisc package. This is to be used with the Cox proportional hazards model only.
- ciwr.mxm concordance index as provided in the rcorr.cens function from the Hmisc package. This is to be used with the Weibull regression model only.
- poisdev.mxm: Poisson regression deviance.
- nbdev.mxm: Negative binomial regression deviance.
- ord_mae.mxm: Ordinal regression mean absolute error.

Usage: metric(predictions, test_target)

Input of modelling functions: train_target: target variable used in the training procedure. sign_data: training set. sign_test: test set.

Modelling functions provide a single vector of predictions obtained by applying the model fit on sign_data and train_target on the sign_test

The modelling functions that are currently supported are:

- glm.mxm: fits a glm for a binomial family (Classification task).
- lm.mxm: fits a linear model model (stats) for the regression task.
- coxph.mxm: fits a cox proportional hazards regression model for the survival task.
- weibreg.mxm: fits a Weibull regression model for the survival task.
- rq.mxm: fits a quantile (median) regression model for the regression task.
- lmrob.mxm: fits a robust linear model model for the regression task.
- pois.mxm: fits a poisson regression model model for the regression task.
- nb.mxm: fits a negative binomial regression model model for the regression task.
- multinom.mxm: fits a multinomial regression model model for the regression task.
- ordinal.mxm: fits an ordinal regression model model for the regression task.
- beta.mxm: fits a beta regression model model for the regression task. The predicted values are transformed into R using the logit transformation. This is so that the "mse.mxm" metric function can be used. In addition, this way the performance can be compared with the regression scenario, where the logit is applied and then a regression model is employed.

Usage: modeler(train_target, sign_data, sign_test)

Note that the Tibshirani and Tibshirani (2009) bias correction method is applied. The procedure will be more automated in the future and more functions will be added. The multithreaded functions have been tested and no error has been detected. However, if you spot any suspicious results please let us know.

Value

A list including:

`cv_results_all`	A list with predictions, performances and signatures for each fold and each SES or MMPC configuration (e.g cv_results_all[[3]]$performances[1] indicates the performance of the 1st fold with the 3d configuration of SES or MMPC). In the case of the multi-threaded functions (cvses.par and cvmmpc.par) this is a list with a matrix. The rows correspond to the folds and the columns to the configurations (pairs of threshold and max_k).

`best_performance`	A numeric value that represents the best average performance.
`BC_best_perf`	A numeric value that represents the bias corrected best average performance.
`best_configuration`	A list that corresponds to the best configuration of SES or MMPC including id, threshold (named 'a') and max_k.

Bear in mind that the values can be extracted with the $ symbol, i.e. this is an S3 class output.

Author(s)

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

References

Tibshirani R.J., and Tibshirani R. (2009). A bias correction for the minimum error rate in cross-validation. The Annals of Applied Statistics 3(2): 822-829.

See Also

`SES`, `CondIndTests`, `testIndFisher`, `testIndLogistic`, `gSquare`, `censIndCR`

Examples

```
set.seed(1234)

# simulate a dataset with continuous data
dataset <- matrix( rnorm(200 * 50), ncol = 50 )
# the target feature is the last column of the dataset as a vector
target <- dataset[, 50]
dataset <- dataset[, -50]

# get 50 percent of the dataset as a train set
train_set <- dataset[1:100, ]
train_target <- target[1:100]

require(hash)
# run a 10 fold CV for the regression task
best_model = cv.ses(target = train_target, dataset = train_set, kfolds = 5, task = "R")

# get the results
best_model$best_configuration
best_model$best_performance

# summary elements of the process. Press tab after each $ to view all the elements and
# choose the one you are intresting in.
# best_model$cv_results_all[[...]]$...
#i.e.
# mse value for the 1st configuration of SES of the 5 fold
abs(best_model$cv_results_all[[1]]$performances[5])
```

```
best_a <- best_model$best_configuration$a
best_max_k <- best_model$best_configuration$max_k
```

`Data simulation from a DAG` *Simulation of data from DAG (directed acyclic graph)*

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

```
rdag(n, p, s, a = 0, m, A = NULL, seed = FALSE)
```

Arguments

`n`	A number indicating the sample size.
`p`	A number indicating the number of nodes (or vectices, or variables).
`s`	A number in $(0, 1)$. This defines somehow the sparseness of the model. It is the probability that a node has an edge.
`a`	A number in $(0, 1)$. The defines the percentage of outliers to be included in the simulated data. If $a = 0$, no outliers are generated.
`m`	A vector equal to the number of nodes. This is the mean vector of the normal distribution from which the data are to be generated. This is used only when $a > 0$ so as to define the mena vector of the multivariate normal from which the outliers will be generated.
`A`	If you already have an an adjacency matrix in mind, plug it in here, otherwise, leave it NULL.
`seed`	If seed is TRUE, the simulated data will always be the same.

Details

In the case where no adjacency matrix is given, an $p \times p$ matrix with zeros everywhere is created. very element below the diagonal is is replaced by random values from a Bernoulli distribution with probability of success equal to s. This is the matrix B. Every value of 1 is replaced by a uniform value in $0.1, 1$. This final matrix is called A. The data are generated from a multivariate normal distribution with a zero mean vector and covariance matrix equal to $\left(\mathbf{I}_p - A\right)^{-1} \left(\mathbf{I}_p - A\right)$, where $\mathbf{I}_p$ is the $p \times p$ identiy matrix. If a is greater than zero, the outliers are generated from a multivariate normal with the same covariance matrix and mean vector the one specified by the user, the argument "m". The flexibility of the outliers is that you cna specifiy outliers in some variables only or in all of them. For example, m=c(0,0,5) introduces outliers in the third variable only, whereas m=c(5,5,5) introduces outliers in all variables. The user is free to decide on the type of outliers to include in the data.

Value

A list including:

`nout`	The number of outliers.
`G`	The adcacency matrix used. If G[i, j] = 1, then G[j, i] = 2 and this means that there is an arrow from j to i.
`A`	The matrix with the with the uniform values in the interval $0.1, 1$.
`x`	The simulated data.

Author(s)

R implementation and documentation: Michail Tsagris <mtsagris@csd.uoc.gr>

References

Colombo, Diego, and Marloes H. Maathuis (2014). Order-independent constraint-based causal structure learning. The Journal of Machine Learning Research 15(1): 3741–3782.

See Also

`pc.skel`, `pc.or`, `mmhc.skel`

Examples

```
y <- rdag(100, 20, 0.2)
x <- y$x
tru <- y$G

mod <- pc.con(x)
b <- pc.or(mod)
plotnetwork(tru)
dev.new()
plotnetwork(b$G)
```

`Forward selection regression`

Variable selection in regression models with forward selection

---

Description

Variable selection in regression models with forward selection

Usage

```
fs.reg(target, dataset, threshold = 0.05, wei = NULL, test = NULL, user_test = NULL,
stopping = "BIC", tol = 2, robust = FALSE, ncores = 1)
```

Arguments

`target`	The class variable. Provide either a string, an integer, a numeric value, a vector, a factor, an ordered factor or a Surv object. See also Details.
`dataset`	The dataset; provide either a data frame or a matrix (columns = variables, rows = samples). In either case, only two cases are avaialble, either all data are continuous, or categorical.
`threshold`	Threshold (suitable values in [0,1]) for asmmmbsing p-values significance. Default value is 0.05.
`wei`	A vector of weights to be used for weighted regression. The default value is NULL. It is not suggested when robust is set to TRUE.
`test`	The regression model to use. Available options are most of the tests for SES and MMPC. The ones NOT available are "gSquare", "censIndER", "testIndMVreg", "testIndClogit", "testIndSpearman" and "testIndFisher".
`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.
`stopping`	The stopping rule. The BIC is always used for all methods. If you have linear regression though you can change this to "adjrsq" and in this case the adjusted R qaured is used.
`tol`	The difference bewtween two successive values of the stopping rule. By default this is is set to 2. If for example, the BIC difference between two succesive models is less than 2, the process stops and the last variable, even though significant does not enter the model.
`robust`	A boolean variable which indicates whether (TRUE) or not (FALSE) to use a robust version of the statistical test if it is available. It takes more time than a non robust version but it is suggested in case of outliers. Default value is FALSE.
`ncores`	How many cores to use. This plays an important role if you have tens of thousands of variables or really large sample sizes and tens of thousands of variables and a regression based test which requires numerical optimisation. In other cases it will not make a difference in the overall time (in fact it can be slower). The parallel computation is used in the first step of the algorithm, where univariate associations are examined, those take place in parallel. We have seen a reduction in time of 50% with 4 cores in comparison to 1 core. Note also, that the amount of reduction is not linear in the number of cores.

Details

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 or the binary logistic regression is used. If the target has two values only, binary logistic regression will be used.
- if target is a ordered factor, the ordered logit regression is used. Hence, if you want to use multinomial or ordinal logistic regression, make sure your target is factor.
- if target is a numerical vector or a matrix with at least two columns (multivariate) linear regression is used.

- if target is discrete numerical (counts), the poisson regression conditional independence test is used. If there are only two values, the binary logistic regression is to be used.
- if target is a Surv object, the Survival conditional independence test is used.

Value

The output of the algorithm is S3 object including:

`mat`	A matrix with the variables and their latest test statistics and p-values.
`info`	A matrix with the selected variables, their p-values and test statistics. Each row corresponds to a model which contains the variables up to that line. The BIC in the last column is the BIC of that model.
`models`	The regression models, one at each step.
`final`	The final regression model.
`ci_test`	The conditional independence test used.
`runtime`	The run time of the algorithm. A numeric vector. The first element is the user time, the second element is the system time and the third element is the elapsed time.

Author(s)

Michail Tsagris

R implementation and documentation: Giorgos Athineou <athineou@csd.uoc.gr> Michail Tsagris <mtsagris@csd.uoc.gr>

See Also

`glm.fsreg`, `lm.fsreg`, `bic.fsreg`, `bic.glm.fsreg`. `CondIndTests`, `MMPC`, `SES`

Examples

```
set.seed(123)

#simulate a dataset with continuous data
dataset <- matrix( runif(1000 * 20, 1, 100), ncol = 20 )

#define a simulated class variable
target <- rt(1000, 10)

a1 <- fs.reg(target, dataset, threshold = 0.05, test = "testIndRQ", stopping = "BIC", tol = 2,
robust = FALSE, ncores = 1 )

y <- survival::Surv(rexp(1000), rep(1,1000) )
a2 <- fs.reg(y, dataset, threshold = 0.05, test = "censIndWR", stopping = "BIC", tol = 2,
robust = FALSE, ncores = 1 )
a3 <- MMPC(target, dataset)

target <- rbinom(1000, 1, 0.6)
b1 <- fs.reg(target, dataset, threshold = 0.05, test = NULL, stopping = "BIC", tol = 2,
robust = FALSE, ncores = 1 )

target <- factor( rbinom(1000, 2, 0.6) )
b2 <- fs.reg(target, dataset, threshold = 0.05, test = NULL, stopping = "BIC", tol = 2,
robust = FALSE, ncores = 1 )
```

`Forward selection with generalised linear regression models`
Variable selection in generalised linear regression models with forward selection

---

Description

Variable selection in generalised linear regression models with forward selection

Usage

```
glm.fsreg(target, dataset, ini = NULL, threshold = 0.05, wei = NULL, tol = 2,
heavy = FALSE, robust = FALSE, ncores = 1)
```

Arguments

`target`	The class variable. Provide either a string, an integer, a numeric value, a vector, a factor, an ordered factor or a Surv object. See also Details.
`dataset`	The dataset; provide either a data frame or a matrix (columns = variables, rows = samples). In either case, only two cases are avaialble, either all data are continuous, or categorical.
`ini`	If you have a set of variables already start with this one. Currently this can only be a matrix with continuous variables. In such cases, the dataset must also contain continuous variables only. Otherwise leave it NULL.
`threshold`	Threshold (suitable values in [0,1]) for assessing the p-values significance. Default value is 0.05.
`wei`	A vector of weights to be used for weighted regression. The default value is NULL. It is not suggested when robust is set to TRUE.
`tol`	The difference bewtween two successive values of the stopping rule. By default this is is set to 2. If for example, the BIC difference between two succesive models is less than 2, the process stops and the last variable, even though significant does not enter the model.
`heavy`	A boolean variable specifying whether heavy computations are required or not. If for exmaple the dataset contains tens of thousands of rows, it is advised to used memory efficient GLMs and hence set this to TRUE.
`robust`	A boolean variable which indicates whether (TRUE) or not (FALSE) to use a robust version of the statistical test if it is available. It takes more time than a non robust version but it is suggested in case of outliers. Default value is FALSE and this is currently supported only for the linear regression.

`ncores`	How many cores to use. This plays an important role if you have tens of thousands of variables or really large sample sizes and tens of thousands of variables and a regression based test which requires numerical optimisation. In other cammmb it will not make a difference in the overall time (in fact it can be slower). The parallel computation is used in the first step of the algorithm, where univariate associations are examined, those take place in parallel. We have seen a reduction in time of 50% with 4 cores in comparison to 1 core. Note also, that the amount of reduction is not linear in the number of cores.

Value

The output of the algorithm is S3 object including:

`runtime`	The run time of the algorithm. A numeric vector. The first element is the user time, the second element is the system time and the third element is the elapsed time.
`mat`	A matrix with the variables and their latest test statistics and p-values.
`info`	A matrix with the selected variables, their p-values and test statistics. Each row corresponds to a model which contains the variables up to that line. The BIC in the last column is the BIC of that model.
`models`	The regression models, one at each step.
`ci_test`	The conditional independence test used.
`final`	The final regression model.

Author(s)

Michail Tsagris

R implementation and documentation: Michail Tsagris <mtsagris@csd.uoc.gr>

See Also

`fs.reg`, `lm.fsreg`, `bic.fsreg`, `bic.glm.fsreg`, `CondIndTests`, `MMPC`, `SES`

Examples

```
set.seed(123)

#simulate a dataset with continuous data
dataset <- matrix( runif(1000 * 50, 1, 100), ncol = 50 )

#define a simulated class variable
target <- rpois(1000, 10)

a <- glm.fsreg(target, dataset, threshold = 0.05, tol = 2, robust = FALSE, ncores = 1 )
b <- MMPC(target, dataset, max_k = 3, threshold = 0.05, test = "testIndPois")
```

Forward selection with linear regression models

Variable selection in linear regression models with forward selection

Description

Variable selection in linear regression models with forward selection

Usage

```
lm.fsreg(target, dataset, ini = NULL, threshold = 0.05, wei = NULL, stopping = "BIC",
tol = 2, robust = FALSE, ncores = 1)
lm.fsreg_heavy(target, dataset, ini = NULL, threshold = 0.05, wei = NULL,
stopping = "BIC", tol = 2, ncores = 1)
```

Arguments

`target`	The class variable. Provide either a string, an integer, a numeric value, a vector, a factor, an ordered factor or a Surv object. See also Details.
`dataset`	The dataset; provide either a data frame or a matrix (columns = variables, rows = samples). In either case, only two cases are avaialble, either all data are continuous, or categorical.
`ini`	If you have a set of variables already start with this one. Currently this can only be a matrix with continuous variables. In such cases, the dataset must also contain continuous variables only. Otherwise leave it NULL.
`threshold`	Threshold (suitable values in [0,1]) for assessing the p-values significance. Default value is 0.05.
`wei`	A vector of weights to be used for weighted regression. The default value is NULL. It is not suggested when robust is set to TRUE.
`stopping`	The stopping rule. The BIC ("BIC") or the adjusted R^2 ("adjrsq") can be used.
`tol`	The difference bewtween two successive values of the stopping rule. By default this is is set to 2. If for example, the BIC difference between two succesive models is less than 2, the process stops and the last variable, even though significant does not enter the model. If the adjusted R^2 is used, the tol should be something like 0.01 or 0.02.
`robust`	A boolean variable which indicates whether (TRUE) or not (FALSE) to use a robust version of the statistical test if it is available. It takes more time than a non robust version but it is suggested in case of outliers. Default value is FALSE.
`ncores`	How many cores to use. This plays an important role if you have tens of thousands of variables or really large sample sizes and tens of thousands of variables and a regression based test which requires numerical optimisation. In other cammmb it will not make a difference in the overall time (in fact it can be slower). The parallel computation is used in the first step of the algorithm, where univariate associations are examined, those take place in parallel. We have seen a reduction in time of 50% with 4 cores in comparison to 1 core. Note also, that the amount of reduction is not linear in the number of cores.

Details

Only linear regression (robust and non robust) is supported from this function. The lm.fsreg_heavy is for the case of many tens of thousands of observations.

Value

The output of the algorithm is S3 object including:

`runtime`	The run time of the algorithm. A numeric vector. The first element is the user time, the second element is the system time and the third element is the elapsed time.
`mat`	A matrix with the variables and their latest test statistics and p-values.
`info`	A matrix with the selected variables, their p-values and test statistics. Each row corresponds to a model which contains the variables up to that line. The BIC in the last column is the BIC of that model.
`models`	The regression models, one at each step.
`ci_test`	The conditional independence test used (either "testIndReg" or "testIndSpeedglm").
`final`	The final regression model.

Author(s)

Michail Tsagris

R implementation and documentation: Michail Tsagris <mtsagris@csd.uoc.gr>

References

Tsamardinos I., Aliferis C. F. and Statnikov, A. (2003). Time and sample efficient discovery of Markov blankets and direct causal relations. In Proceedings of the 9th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 673-678).

See Also

`fs.reg`, `lm.fsreg`, `bic.fsreg`, `bic.glm.fsreg`. `CondIndTests`, `MMPC`, `SES`

Examples

```
set.seed(123)

#simulate a dataset with continuous data
dataset <- matrix( runif(1000 * 50, 1, 100), ncol = 50 )

#define a simulated class variable
target <- 3 * dataset[, 10] + 2 * dataset[, 20] + 3 * dataset[, 30] + rnorm(1000, 0, 5)
a <- lm.fsreg(target, dataset, threshold = 0.05, stopping = "BIC", tol = 2,
robust = FALSE, ncores = 1 )
a <- lm.fsreg_heavy(target, dataset, threshold = 0.05, stopping = "BIC", tol = 2,
ncores = 1 )
b <- fs.reg(target, dataset, threshold = 0.05, test = NULL, stopping = "BIC", tol = 2,
robust = TRUE, ncores = 1 )
```

G-square conditional independence test for discrete data

G-square conditional independence test for discrete data

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, wei = NULL, dataInfo = NULL,
univariateModels = NULL, hash = FALSE, stat_hash = NULL,
pvalue_hash = NULL, robust = FALSE)
```

Arguments

`target`	A numeric vector containing the values of the target variable.
`dataset`	A numeric 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.
`wei`	A vector of weights to be used for weighted regression. The default value is NULL. It is not suggested when robust is set to TRUE.
`dataInfo`	A list object with information on the structure of the data. Default value is NULL.
`univariateModels`	Fast alternative to the hash object for univariate tests. 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.
`robust`	A boolean variable which indicates whether (TRUE) or not (FALSE) to use a robust test. Currently it is not available for this test.

Details

If the number of samples is at least 5 times the number of the parameters to be estimated, the test is performed, otherwise, independence is not rejected (see Tsmardinos et al., 2006, pg. 43)

If hash = TRUE, testIndLogistic requires the arguments 'stat_hash' and 'pvalue_hash' for the hash-based implementation of the statistical 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.

For all the available conditional independence tests that are currently included on the package, please see "?CondIndTests".

Value

A list including:

`pvalue`	A numeric value that represents the logarithm of the generated p-value of the G^2 test (see reference below).
`stat`	A numeric value that represents the generated statistic of the G^2 test (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)

R implementation and documentation: Giorgos Athineou <athineou@csd.uoc.gr>

References

Tsamardinos, Ioannis, Laura E. Brown, and Constantin F. Aliferis. The max-min hill-climbing Bayesian network structure learning algorithm. Machine learning, 2006 65(1): 31–78.

See Also

`SES`, `testIndFisher`, `testIndLogistic`, `censIndCR`, `CondIndTests`

Examples

```
#simulate a dataset with binary data
dataset <- matrix(rbinom(500 * 51, 1, 0.6), ncol = 51)
#initialize binary target
target <- dataset[, 51]
#remove target from the dataset
dataset <- dataset[, -51]
```

```
#run the gSquare conditional independence test for the binary class variable
results <- gSquare(target, dataset, xIndex = 44, csIndex = c(10,20) )
results

#run 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");
target <- as.factor(target)
sesObject2 <- SES(target, dataset, max_k = 3, threshold = 0.05, test = "testIndLogistic");
#print summary of the SES output
summary(sesObject);
summary(sesObject2);
# plot the SES output
# plot(sesObject, mode = "all");
```

Generate random folds for cross-validation

Generate random folds for cross-validation

Description

Random folds for use in a cross validation are generated. There is the option for stratified splitting as well.

Usage

```
generatefolds(target, nfolds = 10, stratified = TRUE, seed = FALSE)
```

Arguments

target	A vector with some data, either continuous or categorical.
nfolds	The number of folds to produce.
stratified	A boolean variable specifying whether stratified random (TRUE) or simple random (FALSE) sampling is to be used when producing the folds.
seed	A boolean variable. If set to TRUE, the folds will always be the same.

Details

I was inspired by the sam command in the package **TunePareto** in order to do the stratified version.

Value

A list with nfolds elements where each elements is a fold containing the indices of the data.

Author(s)

Michail Tsagris

R implementation and documentation: Michail Tsagris <mtsagris@csd.uoc.gr>

See Also

cv.ses

Examples

```
a <- generatefolds(iris[, 5], nfolds = 5, stratified = TRUE)
table(iris[a[[1]], 5])  ## 10 values from each group
```

IAMB backward selection phase

IAMB backward selection phase

Description

IAMB backward selection phase.

Usage

```
iamb.bs(target, dataset, threshold = 0.05, wei = NULL, test = NULL, user_test = NULL,
robust = FALSE)
```

Arguments

target	The class variable. Provide either a string, an integer, a numeric value, a vector, a factor, an ordered factor or a Surv object.
dataset	The dataset; provide either a data frame or a matrix (columns = variables, rows = observations). In either case, only two cases are avaialble, either all data are continuous, or categorical.
threshold	Threshold (suitable values in (0,1)) for assessing p-values significance. Default value is 0.05.
test	The regression model to use. Available options are most of the tests for SES and MMPC. The ones NOT available are "gSquare", "censIndER", "testIndMVreg", "testIndClogit", "testIndSpearman" and "testIndFisher".
wei	A vector of weights to be used for weighted regression. The default value is NULL. It is not suggested when robust is set to TRUE.
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.
robust	A boolean variable which indicates whether (TRUE) or not (FALSE) to use a robust version of the statistical test if it is available. It takes more time than a non robust version but it is suggested in case of outliers. Default value is FALSE.

Details

IAMB stands for Incremental Association Markov Blanket. The algorithm comprises of a forward selection and a modified backward selection process. This functions does the modified backward selection process. In the usual backward selection, among the non significant variabels, the one with the maximum p-value is dropped. So, one variable is removed at every step. In the IAMB backward phase, at aevery step, all non significant variables are removed. This makes it a lot faster.

Value

The output of the algorithm is a list of an S3 object including:

`runtime`	The run time of the algorithm. A numeric vector. The first element is the user time, the second element is the system time and the third element is the elapsed time.
`ci_test`	The conditional independence test used.
`vars`	The selected variables.
`mat`	A matrix with the selected variables and their latest test statistic and p-value. If no variable is selected this is NULL.
`final`	The final regression model.

Author(s)

Michail Tsagris

R implementation and documentation: Michail Tsagris <mtsagris@csd.uoc.gr>

See Also

`glm.fsreg`, `lm.fsreg`, `bic.fsreg`, `bic.glm.fsreg`, `CondIndTests`, `MMPC`, `SES`

Examples

```
set.seed(123)
dataset <- matrix( runif(1000 * 10, 1, 100), ncol = 10 )
target <- rnorm(1000)

a1 <- iamb.bs(target, dataset, threshold = 0.05, test = "testIndRQ")
a2 <- bs.reg(target, dataset, threshold = 0.05, test = "testIndRQ")
```

`IAMB variable selection`

IAMB variable selection

Description

IAMB variable selection.

Usage

```
iamb(target, dataset, threshold = 0.05, wei = NULL, test = NULL, user_test = NULL,
stopping = "BIC", tol = 2, robust = FALSE, ncores = 1, back = "iambbs")
```

Arguments

`target`	The class variable. Provide either a string, an integer, a numeric value, a vector, a factor, an ordered factor or a Surv object.
`dataset`	The dataset; provide either a data frame or a matrix (columns = variables, rows = observations). In either case, only two cases are avaialble, either all data are continuous, or categorical.
`threshold`	Threshold (suitable values in (0,1)) for assessing p-values significance. Default value is 0.05.
`wei`	A vector of weights to be used for weighted regression. The default value is NULL. It is not suggested when robust is set to TRUE.
`test`	The regression model to use. Available options are most of the tests for SES and MMPC. The ones NOT available are "gSquare", "censIndER", "testIndMVreg", "testIndClogit", "testIndSpearman" and "testIndFisher".
`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.
`stopping`	The stopping rule. The BIC is always used for all methods. If you have linear regression though you can change this to "adjrsq" and in this case the adjusted R qaured is used.
`tol`	The difference bewtween two successive values of the stopping rule. By default this is is set to 2. If for example, the BIC difference between two succesive models is less than 2, the process stops and the last variable, even though significant does not enter the model.
`robust`	A boolean variable which indicates whether (TRUE) or not (FALSE) to use a robust version of the statistical test if it is available. It takes more time than a non robust version but it is suggested in case of outliers. Default value is FALSE.
`ncores`	How many cores to use. This plays an important role if you have tens of thousands of variables or really large sample sizes and tens of thousands of variables and a regression based test which requires numerical optimisation. In other cases it will not make a difference in the overall time (in fact it can be slower). The parallel computation is used in the first step of the algorithm, where univariate associations are examined, those take place in parallel. We have seen a reduction in time of 50% with 4 cores in comparison to 1 core. Note also, that the amount of reduction is not linear in the number of cores.
`back`	The backward phase. If this "iambbs" (default value) the IAMB backward phase is performed and hence the IAMB algorithm is completed. If "bs", a simple backward selection phase is performed. This way, the IAMB algorithm is slightly more general.

Details

IAMB stands for Incremental Association Markov Blanket. The algorithm comprises of a forward selection and a modified backward selection process. This functions does the modified backward selection process. In the usual backward selection, among the non singificant variabels, the one with the maximum p-value is dropped. So, one variable is removed at every step. In the IAMB backward phase, at aevery step, all non significant variables are removed. This makes it a lot faster.

Value

The output of the algorithm is a list of an S3 object including:

`vars`	A vector with the selcted variables.
`mod`	The output of the backward phase. In the case of no backward procedure this is the output of the forward phase.
`mess`	If the forward regression returned at most one variable, no backward procedure takes place and a message appears informing the user about this.

Author(s)

Michail Tsagris

R implementation and documentation: Michail Tsagris <mtsagris@csd.uoc.gr>

See Also

`glm.fsreg`, `lm.fsreg`, `bic.fsreg`, `bic.glm.fsreg`, `CondIndTests`, `MMPC`, `SES`

Examples

```
set.seed(123)
dataset <- matrix( runif(100 * 100, 1, 100), ncol = 100 )
target <- rnorm(100)
# heavy = robust = FALSE ; threshold = 0.05 ; wei = user_test = NULL ;
# ncores = 1 ; test = NULL ; stopping = "BIC" ; tol = 0

target = rpois(100, 10)
a1 <- iamb(target, dataset, threshold = 0.05, stopping = "BIC", tol = 0, back = "iambbs")
a2 <- iamb(target, dataset, threshold = 0.05, stopping = "BIC", tol = 0, back = "bs")

#a1 <- iamb(target, dataset, threshold = 0.05, test = "testIndReg",
#stopping = "BIC", tol = 0, back = "iambbs")
#a2 <- iamb(target, dataset, threshold = 0.05, test = "testIndReg",
#stopping = "BIC", tol = 0, back = "bs")
```

`Interactive plot of an (un)directed graph`
Interactive plot of an (un)directed graph

Description

Interactive plot of an (un)directed graph.

Usage

```
plotnetwork(G, titlos)
```

Arguments

`G`	The adjacency matrix as produced from `mmhc.skel`, `pc.skel`, `pc.con` or any other algorithm. This can correspond to an undirected, partially directed or a completely directed graph.
`titlos`	A character argument specifying the title of the graph, for example "PC network".

Details

This visualises the directed graph.

Value

The plot of the directed graph. This is interactive, in the sense that the user can "play" with it. Move the nodes, zoom it, strectch it etc.

Author(s)

Michail Tsagris

R implementation and documentation: Giorgos Athineou <athineou@csd.uoc.gr> and Michail Tsagris <mtsagris@csd.uoc.gr>

See Also

`mmhc.skel`, `nei`, `pc.skel`, `mb`

Examples

```
# simulate a dataset with continuous data
dataset <- matrix(runif(1000 * 20, 1, 100), nrow = 1000 )
a <- mmhc.skel(dataset, max_k = 3, threshold = 0.05, test = "testIndFisher",
rob = FALSE, nc = 1)
plotnetwork(a$G)
plotnetwork(a$G, titlos = "DAG skeleton")
```

`mammpc.output-class` *Class* `"mammpc.output"`

Description

mammpc. output object class.

Objects from the Class

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

Slots

`selectedVars`: Object of class `"numeric"`

`selectedVarsOrder`: Object of class `"numeric"`

`hashObject`: Object of class `"list"`

`pvalues`: Object of class `"numeric"`

`stats`: Object of class `"numeric"`

`univ`: Object of class `"list"`

`max_k`: Object of class `"numeric"`

`threshold`: Object of class `"numeric"`

`test`: Object of class `"character"`

`runtime`: Object of class `"proc_time"`

`rob`: Object of class `"logical"`

Methods

summary `summary(object = "mammpc.output")`: Generic function for summarizing the results of the meta analytic MMPC output

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

Author(s)

Giorgos Athineou <athineou@csd.uoc.gr>

See Also

`ma.mmpc`, `ma.ses`

Examples

```
showClass("mammpc.output")
```

`Many simple beta regressions` *Many simple beta regressions.*

Description

Many simple beta regressions.

Usage

```
beta.regs(target, dataset, wei = NULL, logged = FALSE, ncores = 1)
```

Arguments

`target`	The target (dependent) variable. It must be a numerical vector with integers.
`dataset`	The indendent variable(s). It can be a vector, a matrix or a dataframe with continuous only variables, a data frame with mixed or only categorical variables.
`wei`	A vector of weights to be used for weighted regression. The default value is NULL.
`logged`	A boolean variable; it will return the logarithm of the pvalue if set to TRUE.
`ncores`	The number of cores to use. The default value is 1.

Details

Many simple beta regressions are fitted.

Value

A matrix with the test statistic values, their relevant (logged) p-values and the BIC values.

Author(s)

Michail Tsagris

R implementation and documentation: Michail Tsagris <mtsagris@csd.uoc.gr>

References

Ferrari S.L.P. and Cribari-Neto F. (2004). Beta Regression for Modelling Rates and Proportions. Journal of Applied Statistics, 31(7): 799-815.

See Also

`beta.mod`, `testIndBeta`, `reg.fit`, `ridge.reg`

Examples

```
y <- rbeta(500, 5, 3)
x <- matrix( rnorm(500 * 20), ncol = 20)
a <- beta.regs(y, x)
```

`Many simple zero inflated Poisson regressions`
Many simple zero inflated Poisson regressions.

Description

Many simple zero inflated Poisson regressions.

Usage

```
zip.regs(target, dataset, wei = NULL, logged = FALSE, ncores = 1)
```

Arguments

`target`	The target (dependent) variable. It must be a numerical vector with integers.
`dataset`	The indendent variable(s). It can be a vector, a matrix or a dataframe with continuous only variables, a data frame with mixed or only categorical variables.
`wei`	A vector of weights to be used for weighted regression. The default value is NULL.
`logged`	A boolean variable; it will return the logarithm of the pvalue if set to TRUE.
`ncores`	The number of cores to use. The default value is 1.

Details

Many simple zero inflated Poisson regressions are fitted.

Value

A matrix with the test statistic values, their relevant (logged) p-values and the BIC values.

Author(s)

Michail Tsagris

R implementation and documentation: Michail Tsagris <mtsagris@csd.uoc.gr>

References

Lambert D. (1992). Zero-inflated Poisson regression, with an application to defects in manufacturing. Technometrics, 34(1):1-14.

See Also

zip.mod, testIndZIP, reg.fit, ridge.reg

Examples

```
y <- rpois(1000, 3)
x <- matrix( rnorm(1000 * 20), ncol = 20)
y[1:100] <- 0
a <- zip.regs(y, x)
```

Markov Blanket of a node in a directed graph

Returns and plots, if asked, the Markov blanket of a node (or variable)

Description

Returns and plots, if asked, the Markov blanket of a node (or variable).

Usage

```
mb(G, node, graph = FALSE)
```

Arguments

`G`	The graph matrix as produced from pc.or or any other algorithm which produces directed graphs.
`node`	A vector with one or more numbers indicating the seleted node(s) (or variable(s)).
`graph`	A boolean variable. If TRUE the relevant graph will appear (if the Markov blanket is the non emty set).

Details

This is a way to see the network for some given nodes. It is useful if you have many nodes and the whole network is a bit difficult to see clearly. Bear in mind that the values can be extracted with the $ symbol, i.e. this is an S3 class output.

Value

`parents`	The parents of the node of interest.
`children`	The children of the node of interest.
`spouses`	The spouses of the node of interest. These are the other parents of the children of the node of interest.
`relatives`	Nodes which are connected with the node of interest, but it is not known whether they are parents or children. The edge between them is undirected.
`markov.blanket`	The Markov blanket of the node of interest. The collection of all the previous.

Author(s)

Michail Tsagris

R implementation and documentation: Michail Tsagris <mtsagris@csd.uoc.gr>

See Also

`plotnetwork`, `nei`, `pc.or`

Examples

```
# simulate a dataset with continuous data
# simulate a dataset with continuous data
y <- rdag(1000, 10, 0.3)
tru <- y$G
x <- y$x
mod <- pc.con(x)
G <- pc.or(mod)$G
plotnetwork(G)
dev.new()
mb(G, 8, graph = TRUE)
```

`mases.output-class` *Class* `"mases.output"`

Description

Meta analytic SES output object class.

Objects from the Class

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

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"`

`univ`: Object of class `"list"`

`max_k`: Object of class `"numeric"`

`threshold`: Object of class `"numeric"`

`runtime`: Object of class `"proc_time"`

`test`: Object of class `"character"`

`rob`: Object of class `"logical"`

Methods

summary `summary(object = "mases.output")`: Generic function for summarizing the results of the meta analytic SES output

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

Author(s)

Giorgos Athineou <athineou@csd.uoc.gr>

See Also

`ma.ses`, `ma.mmpc`

Examples

```
showClass("mases.output")
```

`MMPC solution paths for many combinations of hyper-parameters`
MMPC solution paths for many combinations of hyper-parameters

---

Description

MMPC solution paths for many combinations of hyper-parameters.

Usage

```
mmpc.path(target, dataset, wei = NULL, max_ks = NULL, thresholds = NULL, test = NULL,
user_test = NULL, robust = FALSE, ncores = 1)
```

Arguments

`target`	The class variable. Provide either a string, an integer, a numeric 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).
`wei`	A vector of weights to be used for weighted regression. The default value is NULL.
`max_ks`	A vector of possible max_k values. Can be a number as well, but this does not really make sense to do. If nothing is given, the values max_k=3 and max_k=2 are used by default.

`thresholds`	A vector of possible threshold values. Can be a number as well, but this does not really make sense to do. If nothing is given, the values (0.1, 0.05, 0.01) are used by default.
`test`	The conditional independence test to use. Default value is NULL. See also `CondIndTests`.
`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.
`robust`	A boolean variable which indicates whether (TRUE) or not (FALSE) to use a robust version of the statistical test if it is available. It takes more time than a non robust version but it is suggested in case of outliers. Default value is FALSE.
`ncores`	How many cores to use. This plays an important role if you have tens of thousands of variables or really large sample sizes and tens of thousands of variables and a regression based test which requires numerical optimisation. In other cases it will not make a difference in the overall time (in fact it can be slower). The parallel computation is used in the first step of the algorithm, where univariate associations are examined, those take place in parallel. We have seen a reduction in time of 50% with 4 cores in comparison to 1 core. Note also, that the amount of reduction is not linear in the number of cores. This argument is used only in the first run of MMPC and for the univariate associations only and the results are stored (hashed). In the enxt runs of MMPC the results are used (cashed) and so the process is faster.

Details

For different combinations of the hyper-parameters, max_k and the significance level (threshold or alpha) the MMPC algorith is run.

Value

The output of the algorithm is an object of the class 'SESoutput' for SES or 'MMPCoutput' for MMPC including:

`bic`	A matrix with the BIC values of the final fitted model based on the selected variables identified by each configuration, combination of the hyper-parameters.
`size`	A matrix with the legnth of the selected variables identified by each configuration of MMPC.
`variables`	A list containing the variables from each configuration of MMPC
`runtime`	The run time of the algorithm. A numeric vector. The first element is the user time, the second element is the system time and the third element is the elapsed time.

Author(s)

Ioannis Tsamardinos, Vincenzo Lagani

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

Tsamardinos, Brown and Aliferis (2006). The max-min hill-climbing Bayesian network structure learning algorithm. Machine learning, 65(1): 31-78.

See Also

`CondIndTests`, `cv.ses`

Examples

```
set.seed(123)
require(hash)
# simulate a dataset with continuous data
dataset <- matrix(runif(1000 * 101, 1, 100), nrow = 1000 )
#the target feature is the last column of the dataset as a vector
target <- dataset[, 101]
dataset <- dataset[, -101]

a <- mmpc.path(target, dataset, max_ks = NULL, thresholds = NULL, test = NULL,
user_test = NULL, robust = FALSE, ncores = 1)
```

`MMPC.temporal.output-class`

Class "`MMPC.temporal.output`"

Description

MMPC.temporal output object class.

Objects from the Class

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

Slots

`selectedVars`**:** Object of class "`numeric`"

`selectedVarsOrder`**:** Object of class "`numeric`"

`hashObject`**:** Object of class "`list`"

`pvalues`**:** Object of class "`numeric`"

`stats`**:** Object of class "`numeric`"

`univ`**:** Object of class "`list`"

`max_k`**:** Object of class "`numeric`" `threshold`: Object of class "`numeric`"

`runtime`: Object of class "`proc_time`"

`test`: Object of class "`character`"

`slope`: Object of class "`logical`"

Methods

summary `summary(object = "MMPC.temporal.output")`: Generic function for summarizing the results of the MMPC.temporal. output

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

Author(s)

Giorgos Athineou <athineou@csd.uoc.gr>

See Also

`MMPC.temporal`, `SES.temporal`

Examples

```
showClass("MMPC.temporal.output")
```

`MMPCoutput-class` *Class* "`MMPCoutput`"

Description

MMPC output object class.

Objects from the Class

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

Slots

`selectedVars`: Object of class "`numeric`"

`selectedVarsOrder`: Object of class "`numeric`"

`hashObject`: Object of class "`list`"

`pvalues`: Object of class "`numeric`"

`stats`: Object of class "`numeric`"

`univ`: Object of class "`list`"

`max_k`: Object of class "`numeric`"

`threshold`**:** Object of class `"numeric"`

`runtime`**:** Object of class `"proc_time"`

`test`**:** Object of class `"character"`

`rob`**:** Object of class `"logical"`

Methods

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

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

Author(s)

Giorgos Athineou <athineou@csd.uoc.gr>

See Also

`MMPC`, `SES`

Examples

```
showClass("MMPCoutput")
```

`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.

Functions

- InternalSES(...)
- InternalMMPC(...)
- Internalmases(...)
- Internalmammpc(...)
- IdentifyEquivalence(...)
- IdentifyEquivalence.ma(...)
- apply_ideq(...)

- apply_ideq.ma(...)
- compare_p_values(...)
- identifyTheEquivalent(...)
- identifyTheEquivalent.ma(...)
- max_min_assoc(...)
- max_min_assoc.ma(...)
- min_assoc(...)
- min_assoc.ma(...)
- univariateScore(...)
- univariateScore.ma(...)
- cat.ci(...)
- condi.perm(...)
- InternalSES.temporal(...)
- InternalMMPC.temporal(...)
- IdentifyEquivalence.temporal(...)
- apply_ideq.temporal(...)
- identifyTheEquivalent.temporal(...)
- max_min_assoc.temporal(...)
- min_assoc.temporal(...)
- univariateScore.temporal(...)
- is.sepset(...)
- lm.fsreg_2(...)
- glm.fsreg_2(...)
- cvses.par(...)
- cvmmpc.par(...)
- dag_to_eg(...)
- topological_sort(...)
- nchoosek(...)
- R0(...)
- R1(...)
- R2(...)
- R3(...)
- is.sepset(...)
- regbeta(...)
- regbetawei(...)
- betamle.wei(...)
- regzip(...)

- regzipawei(...)
- zipmle.wei(...)
- zipwei(...)
- bic.betafsreg(...)
- bic.zipfsreg(...)
- beta.fsreg(...)
- zip.fsreg(...)
- beta.bsreg(...)
- zip.bsreg(...)
- vara(...)
- iamb.betabs(...)
- iamb.zipbs(...)
- iamb.glmbs(...)
- internaliamb.binombs(...)
- internaliamb.poisbs(...)
- internaliamb.lmbs(...)
- lm.fsreg_2.heavy(...)

`Neighbours of nodes in an undirected graph`
Returns and plots, if asked, the node(s) and their neighbour(s), if there are any.

Description

Returns and plots, if asked, the node(s) and their neighbour(s) of one or more nodes (if there are any).

Usage

```
nei(G, node, graph = TRUE)
```

Arguments

`G`	The adjacency matrix of an undirected graph as produced by `mmhc.skel`, `pc.skel` or any other algorithm.
`node`	A vector with one or more numbers indicating the seleted node(s) (or variable(s)).
`graph`	A boolean variable. If TRUE the relevane graph will appear (if there are neighbours).

Details

This is a way to see the network for some given nodes. It is useful if you have many nodes and the whole network is a bit difficult to see clearly.

Value

A list object called "geit" containing the neighbours of the node(s). If there are no neighbours a message appears and no plot is presented. If the "graph" argument is set to TRUE and there are neighbours, a plot will appear.

Bear in mind that the values can be extracted with the $ symbol, i.e. this is an S3 class output.

Author(s)

Michail Tsagris

R implementation and documentation: Giorgos Athineou <athineou@csd.uoc.gr> and Michail Tsagris <mtsagris@csd.uoc.gr>

See Also

`mmhc.skel`, `SES`, `MMPC`

Examples

```
# simulate a dataset with continuous data
set.seed(1234)
dataset <- matrix(runif(1000 * 20, 1, 100), nrow = 1000 )
G <- pc.con(dataset)$G
plotnetwork(G)
dev.new()
nei( G, c(3, 4) )
dev.new()
nei( G, c(1, 3) )
```

Orientation rules for the PC algorithm

The orientations part of the PC algorithm.

Description

The function takes the outcome of the PC algorithm, as produced by `pc.skel` or `pc.con` and performes the 4 orientation rules. A graph is also possible to visualize.

Usage

```
pc.or(mod, graph = FALSE)
```

Arguments

`mod`	An object with the results of the PC algorithm, as produced by `pc.skel` or `pc.con`.
`graph`	Boolean that indicates whether or not to generate a plot with the graph. Package RgraphViz is required.

Details

After having calculated the skeleton of the PC algorithm one may wants to perform orientations, leading to causal relationships. The rules as stated in Spirtes, Glymour and Scheines (2001) are

1. **Rule 1**. For each triple of vertices X, Y, Z such that the pair X, Y and the pair Y, Z are each adjacent in C but the pair X, Z are not adjacent in C, orient X - Y - Z as X -> Y <- Z if and only if Y is not in Sepset(X, Z).
2. **Rule 2**. If A -> B, B and C are adjacent, A and C are not adjacent, and there is no arrowhead at B, then orient B - C as B -> C.
3. **Rule 3**. If there is a directed path from A to B, and an edge between A and B, then orient A - B as A -> B.
4. **Rule 4**. This was added by Zhang (2008). If A -> B <- C, A - D - C, A and C are not adjacent, and D - B, then orient D - B as D -> B.

The first rule is applied once. Rules 2-4 are applied repeatedly until no more edges can be oriented.

Value

A list including:

`Gini`	The initial adjacency matrix, no orientations. This is the matrix produced by `pc.skel` or `pc.con`.
`G`	The final adjaceny matrix with the orientations. If G[i, j] = 2 then G[j, i] = 3. This means that there is an arrow from node i to node j. If G[i, j] = G[j, i] = 0; there is no edge between nodes i and j. If G[i, j] = G[j, i] = 1; there is an (undirected) edge between nodes i and j.
`runtime`	The run time of the algorithm. A numeric vector. The first element is the user time, the second element is the system time and the third element is the elapsed time.

Author(s)

Michail Tsagris

R implementation and documentation: Michail Tsagris <mtsagris@csd.uoc.gr>

References

Spirtes P., Glymour C. and Scheines R. (2001). Causation, Prediction, and Search. The MIT Press, Cambridge, MA, USA, 3nd edition.

Zhang, Jiji. (2008). On the completeness of orientation rules for causal discovery in the presence of latent confounders and selection bias. Artificial Intelligence 172(16): 1873–1896.

See Also

pc.con, pc.skel, mmhc.skel, mb

Examples

```
# simulate a dataset with continuous data
y <- rdag(1000, 15, 0.3)
tru <- y$G
x <- y$x
mod <- pc.con(x)
mod$runtime

b <- pc.or(mod)
plotnetwork(tru)
dev.new()
plotnetwork(b$G)

dev.off()
plotnetwork( dag2eg(tru) )  ## essential graph
dev.new()
plotnetwork(b$G)
```

`Partial correlation between two variables`
Partial correlation

Description

Partial correlation between two variables when a correlation matrix is given.

Usage

```
partialcor(R, indx, indy, indz)
```

Arguments

R	A correlation matrix.
indx	The index of the first variable whose conditional correlation is to estimated.
indy	The index of the second variable whose conditional correlation is to estimated.
indz	The index of the conditioning variables.

Details

Given a correlation matrix the function will caclulate the partial correlation between variables indx and indy conditioning on variable(s) indz.

Value

A number, the partial correlation coefficient.

Author(s)

Michail Tsagris

R implementation and documentation: Michail Tsagris <mtsagris@csd.uoc.gr>

See Also

`testIndFisher`, `testIndSpearman`, `permcor`, `pc.con`

Examples

```
r <- cor( iris[, 1:4] )
partialcor(r, 1, 2, 0)
r[1, 2]  ## the same as above

y = as.vector( iris[, 1] )
x = as.vector( iris[, 2] )
z = as.vector( iris[, 3] )
e1 = resid( lm(y ~ z) )
e2 = resid( lm(x ~ z) )
cor(e1, e2)
partialcor(r, 1,2, 3)
```

Permutation based p-value for the Pearson correlation coefficient

Permutation based p-value for the Pearson correlation coefficient

---

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

```
permcor(x, R = 999)
permcorrels(y, x, R = 999)
```

Arguments

`x`	For the case of "permcor" this is a matrix with two columns, two continuous variables. In the case of "permcorrels" this is a matrix with many variables.
`y`	A vector whose length is equal to the number of rows of x.
`R`	The number of permutations to be conducted; set to 999 by default.

Details

This is a computational non parametric correlation coefficient test and is advised to be used when a small sample size is available. If you want to use the Spearman correlation instead, simply provide the ranks of x or of y and x.

Value

For the case of "permcor" a vector consisting of two values, the Pearson correlation and the permutation based p-value. For the "permcorrels" a vector with three values, the Pearson correlation, the test statistic value and the permutation based p-value.

Author(s)

Michail Tsagris

R implementation and documentation: Michail Tsagris <mtsagris@csd.uoc.gr>

References

Legendre Pierre (2000). Comparison of permutation methods for the partial correlation and partial Mantel tests. Journal of Statistical Computation and Simulation 67(1):37-73.

See Also

`pc.skel`, `testIndSpearman`, `testIndFisher`, `SES`, `CondIndTests`

Examples

```
permcor(iris[, 1:2])
permcor(iris[, 1:2], R = 9999)
x <- matrix(rnorm(50 * 2000), ncol = 2000)
a <- permcorrels(iris[1:50, 1], x)
```

`Plot of longitudinal data`

Plot of longitudinal data

Description

Plot of longitudinal data.

Usage

```
tc.plot(target, tp, id, type = "l", ylab = "Values", xlab = "Time points",
       col = 2, lwd = 1, lty = 2, pch = 1)
```

Arguments

`target`	A numerical vector with the longitudinal data.
`tp`	The time points. It can either be a vector with length either equal to the number of time points or equal to the legnth of the target.
`id`	A numerical vector specifying the subjects. It can either be a vector with length either equal to the number of subjects or equal to the legnth of the target.
`type`	This is a graphical parameter. You can have lines "l" everywhere or lines with points at each time point "p".
`ylab`	This is a graphical parameter. The label on the y axis.
`xlab`	This is a graphical parameter. The label on the x axis.
`col`	This is a graphical parameter. The color of the lines.
`lwd`	This is a graphical parameter. The thickness of the lines.
`lty`	This is a graphical parameter. The type of line, e.g. dashed, dotted, etc.
`pch`	This is a graphical parameter. If the type is "b", then you can specify if you want different signs, for example circles, crosses, diamonds etc.

Details

The data must be longitudinal (the same subject measured multiple times at different time points) and for one variable only. For the graphical parameters see `plot` or `par`.

Value

A plot with the longitudinal data over time.

Author(s)

Michail Tsagris

R implementation and documentation: Giorgos Athineou <athineou@csd.uoc.gr> Michail Tsagris <mtsagris@csd.uoc.gr>

References

Tsamardinos I., Aliferis C. F. and Statnikov, A. (2003). Time and sample efficient discovery of Markov blankets and direct causal relations. In Proceedings of the 9th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 673-678).

See Also

`testIndGLMM`, `SES.temporal`

Examples

```
## require(lme4)
## data(sleepstudy)
## attach(sleepstudy)
## tc.plot(Reaction, Days, Subject)
## tc.plot(Reaction, Days, Subject, type = "b")
```

`Regression models based on SES and MMPC outputs`
Regression model(s) obtained from SES or MMPC

Description

One or more regression models obtained from SES or MMPC, are returned.

Usage

```
ses.model(target, dataset, wei = NULL, sesObject, nsignat = 1, test = NULL)

mmpc.model(target, dataset, wei = NULL, mmpcObject, test = NULL)
```

Arguments

`target`	A numeric vector containing the values of the target variable. If the values are proportions or percentages, i.e. strictly within 0 and 1 they are mapped into R using log(target/(1 - target)). It can also discrete data, binary data (as factor), nominal or ordinal data (as factor). In contrast to SES, no position of the target variable in the dataset is accepted. The target must be a numerical vector.
`dataset`	A numeric matrix containing the variables. Rows are samples and columns are features. If you have categorical variables, this should be a data frame.
`wei`	A vector of weights to be used for weighted regression. The default value is NULL. We suggest not to use weights if you choose testIndReg and robust = TRUE (robust regression via M estimation)
`sesObject`	An object with the results of a SES run.
`mmpcObject`	An object with the results of an MMPC run.
`nsignat`	How many signatures to use. If nsignat = 1 (default value) the first set of variables will be used for the model. If you want more, then specify the nubmer of signatures you want. If you want the models based on all signatures, specify "all". If you put a number which is higher than the number of signatures, all models will be returned.
`test`	If you know the test used in SES put it here, otherwise leave it NULL. It will take this information from the SEs object. If you used a robust version of a test (wherever possible), robust model(s) will be created.

Details

This command is useful if you want to see all models and check for example their fitting ability, MSE in linear models for exmaple.

Value

A list including:

`mod`	Depending on the number of signatures requested, one or models will be returned.
`signature`	A matrix (or just one vector if one signature only) with the variables of each signature, along with the BIC of the corresponding regression model.

Author(s)

Michail Tsagris

R implementation and documentation: Giorgos Athineou <athineou@csd.uoc.gr> and Michail Tsagris <mtsagris@csd.uoc.gr>

References

Aitchison J. (1986). The Statistical Analysis of Compositional Data, Chapman & Hall; reprinted in 2003, with additional material, by The Blackburn Press.

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

Draper, N.R. and Smith H. (1988). Applied regression analysis. New York, Wiley, 3rd edition.

Ferrari S.L.P. and Cribari-Neto F. (2004). Beta Regression for Modelling Rates and Proportions. Journal of Applied Statistics, 31(7): 799-815.

Gutenbrunner C., Jureckova J., Koenker R. and Portnoy S. (1993). Tests of Linear Hypothesis based on Regression Rank Scores, Journal of NonParametric Statistics 2, 307-331.

Joseph M.H. (2011). Negative Binomial Regression. Cambridge University Press, 2nd edition.

Koenker R.W. (2005). Quantile Regression, Cambridge University Press.

Lagani V., Kortas G. and Tsamardinos I. (2013). Biomarker signature identification in "omics" with multiclass outcome. Computational and Structural Biotechnology Journal, 6(7): 1-7.

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

Lambert, Diane (1992). Zero-inflated Poisson regression, with an application to defects in manufacturing. Technometrics 34(1)1: 1-14.

Mardia K.V., Kent J.T. and Bibby J.M. (1979). Multivariate Analysis, Academic Press, New York, USA.

Maronna R.D. Yohai M.V. (2006). Robust Statistics, Theory and Methods. Wiley.

McCullagh P., and Nelder J.A. (1989). Generalized linear models. CRC press, USA, 2nd edition.

See Also

`SES`, `MMPC`, `cv.ses`, `cv.mmpc`

Examples

```
# simulate a dataset with continuous data
dataset <- matrix(runif(1000 * 101, 1, 100), nrow = 1000 )
#the target feature is the last column of the dataset as a vector
target <- dataset[, 101]
dataset <- dataset[, -101]
sesObject <- SES(target , dataset , max_k=3 , threshold = 0.05);
ses.model(target, dataset, sesObject = sesObject, nsignat = 1, test = NULL)
ses.model(target, dataset, sesObject = sesObject, nsignat = 40, test = NULL)
mmpcObject <- MMPC(target , dataset , max_k=3 , threshold = 0.05);
mmpc.model(target, dataset, mmpcObject = mmpcObject, test = NULL)
```

`Regression models fitting` *Regression modelling*

Description

Generic regression modelling function.

Usage

```
reg.fit(y, dataset, event = NULL, reps = NULL, group = NULL, slopes = FALSE,
reml = FALSE, model = NULL, robust = FALSE, wei = NULL, xnew = NULL)
```

Arguments

`y`	The target (dependent) variable. It can be a numerical variable, factor, ordinal factor, percentages, matrix, or time to event. If the values are proportions or percentages, i.e. strictly within 0 and 1 they are mapped into R using log(target/(1 - target)). If they are compositional data the additive log-ratio (multivariate logit) transformation is aplied beforehand.
`dataset`	The indendent variable(s). It can be a vector, a matrix or a dataframe with continuous only variables, a data frame with mixed or only categorical variables.
`event`	This is NULL unless you have time to event data (survival regression).
`reps`	This is NULL unless you have time measurements (longitudinal data).
`group`	This is NULL unless you have grouped (or clustered) data or longitudinal data (is the latter case the arugment reps is required).
`slopes`	This is for the longitudinal data only, TRUE or FALSE. Should random slopes be added or not?
`reml`	This is for the longitudinal or grouped data only, TRUE or FALSE. If TRUE, REML will be used, otherwise ML will be used.

`model`	The type of model you want to use. It can be specified by the user or left NULL, if other correct arguments have been passed. Poissible values (apart from NULL) are: "gaussian" (default), "binary", "binomial", "multinomial", "poisson", "ordinal", "Cox", "weibull", "exponential", "zip", "beta", "median", "negbin", "longitudinal" and "grouped". The "zip" means that the zero part is constant, the variables are not associated with the excessive zeros. The value "grouped" refers to grouped data, but this does not have to be given if the argument "group" is given, but not the argument "reps. The "binomial" is when you have the number of successes and also the number of trials.
`robust`	A boolean variable. If TRUE robust models will be returned. Currently this is supported by the "gaussian" model only.
`wei`	A vector of weights to be used for weighted regression. The default value is NULL. We suggest not to use weights if you choose testIndReg and robust = TRUE (robust regression via M estimation)
`xnew`	If you have new data whose target values you want to predict put it here, otherwise leave it blank.

Details

This is a generic regression function, which offers prediction as well. It is important that you pass the arguments with their names, for example if you have time to event data, write "event = ..." and not just put your event variable. This will avoid confusion. For the mixed models you need to specify the relevant arguments, "slopes", "reps", "reml" and "group"

Value

A list including:

`mod`	The fitted model.
`pred`	If you have new data the predicted values of the target (dependent) variable.

Author(s)

Michail Tsagris

R implementation and documentation: Michail Tsagris <mtsagris@csd.uoc.gr>

References

Almost the same as in `CondIndTests`.

See Also

`ridge.reg`, `ses.model`, `mmpc.model`

Examples

```
#simulate a dataset with continuous data
dataset <- matrix(runif(100 * 10, 1, 100), nrow = 100 )
#the target feature is the last column of the dataset as a vector
target <- dataset[, 1]
dataset <- dataset[, -1]
a <- reg.fit(target, dataset)
```

Ridge regression *Ridge regression*

Description

Regularisation via ridge regression is performed.

Usage

```
ridge.reg(target, dataset, lambda, B = 1, newdata = NULL)
```

Arguments

`target`	A numeric vector containing the values of the target variable. If the values are proportions or percentages, i.e. strictly within 0 and 1 they are mapped into R using log(target/(1 - target)).
`dataset`	A numeric matrix containing the variables. Rows are samples and columns are features.
`lambda`	The value of the regularisation parameter λ.
`B`	Number of bootstraps. If B = 1 no bootstrap is performed and no standard error for the regression coefficients is returned.
`newdata`	If you have new data and want to predict the value of the target put them here, otherwise, leave it NULL.

Details

There is also the lm.ridge command in MASS library if you are interested in ridge regression.

Value

A list including:

`beta`	The regression coefficients if no bootstrap is performed. If bootstrap is performed their standard error appears as well.
`seb`	The standard erorr of the regression coefficients. If bootstrap is performed their bootstrap estimated standard error appears.
`est`	The fitted values if no new data are available. If you have used new data these will be the predicted target values.

Author(s)

Michail Tsagris

R implementation and documentation: Michail Tsagris <mtsagris@csd.uoc.gr>

References

Hoerl A.E. and R.W. Kennard (1970). Ridge regression: Biased estimation for nonorthogonal problems. Technometrics, 12(1): 55-67.

Brown P. J. (1994). Measurement, Regression and Calibration. Oxford Science Publications.

See Also

`ridgereg.cv`

Examples

```
#simulate a dataset with continuous data
dataset <- matrix(runif(100 * 50, 1, 100), nrow = 100 )
#the target feature is the last column of the dataset as a vector
target <- dataset[, 10]
dataset <- dataset[, -10]
a0 <- ridge.reg(target, dataset, lambda = 0, B = 1, newdata = NULL)
a1 <- ridge.reg(target, dataset, lambda = 0.5, B = 1, newdata = NULL)
a2 <- ridge.reg(target, dataset, lambda = 0.5, B = 100, newdata = NULL)
```

`Ridge regression coefficients plot`
Ridge regression

Description

A plot of the regularised parameters is shown.

Usage

```
ridge.plot(target, dataset, lambda = seq(0, 5, by = 0.1) )
```

Arguments

`target`	A numeric vector containing the values of the target variable. If the values are proportions or percentages, i.e. strictly within 0 and 1 they are mapped into R using log(target/(1 - target)). In any case, they must be continuous only.
`dataset`	A numeric matrix containing the continuous variables. Rows are samples and columns are features.
`lambda`	A grid of values of the regularisation parameter λ.

Details

For every value of λ the coefficients are obtained. They are plotted versus the λ values.

Value

A plot with the values of the coefficients as a function of λ.

Author(s)

Michail Tsagris

R implementation and documentation: Giorgos Athineou <athineou@csd.uoc.gr>, Vincenzo Lagani <vlagani@csd.uoc.gr> and Michail Tsagris <mtsagris@csd.uoc.gr>

References

Hoerl A.E. and R.W. Kennard (1970). Ridge regression: Biased estimation for nonorthogonal problems. Technometrics, 12(1): 55-67.

Brown P. J. (1994). Measurement, Regression and Calibration. Oxford Science Publications.

See Also

`ridge.reg`, `ridgereg.cv`

Examples

```
#simulate a dataset with continuous data
dataset <- matrix(runif(300 * 20, 1, 20), nrow = 300 )
#the target feature is the last column of the dataset as a vector
target <- dataset[, 20]
dataset <- dataset[, -20]
ridge.plot(target, dataset)
```

`ROC and area under the curve` *ROC and area under the curve*

Description

Receiver operating curve and area under the curve.

Usage

```
auc(group, preds, roc = FALSE, cutoffs = NULL)
```

Arguments

`group`	The true labels, either a factor or a numerical vector with two numbers only.
`preds`	The predicted values of each group.
`roc`	If you want the ROC to appear set it to TRUE.
`cutoffs`	If you provide a vector with decreasing numbers from 1 to 0 that will be used for the ROC, otherwise, the values from 1 to 0 with a step equal to -0.01 will be used.

Details

The ara under the curve is returned. The user has the option of getting the receiver operating curve as well.

Value

If the roc is set to FALSE, the area under the curve is returned only. Otherwise a list including

`cutoffs`	The cutoff values.
`sensitivity`	The sensitivity values for each cutoff value.
`specificity`	The specificity value for each cutoff value.
`youden`	The Youden index which is defined as the maximum value of sensitivity - specificity + 1.
`auc`	The area under the curve.

Author(s)

Michail Tsagris

R implementation and documentation: Michail Tsagris <mtsagris@csd.uoc.gr>

See Also

`SES`, `MMPC`, `testIndLogistic`

Examples

```
# simulate a dataset with continuous data
g <- rbinom(150, 1, 0.6)
f <- rnorm(150)
auc(g, f, roc = TRUE)
```

`SES.temporal.output-class` *Class* `"SES.temporal.output"`

Description

SES.temporal output object class.

Objects from the Class

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

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"`

`univ`: Object of class `"list"`

`max_k`: Object of class `"numeric"`

`threshold`: Object of class `"numeric"`

`runtime`: Object of class `"proc_time"`

`test`: Object of class `"character"`

`slope`: Object of class `"logical"`

Methods

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

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

Author(s)

Giorgos Athineou <athineou@csd.uoc.gr>

See Also

`MMPC.temporal`, `SES.temporal`

Examples

```
showClass("SES.temporal.output")
```

`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"`

`univ`**:** Object of class `"list"`

`max_k`**:** Object of class `"numeric"`

`threshold`**:** Object of class `"numeric"`

`runtime`**:** Object of class `"proc_time"`

`test`**:** Object of class `"character"`

`rob`**:** Object of class `"logical"`

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@csd.uoc.gr>

See Also

`SES`

Examples

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

`Skeleton of the max-min hill-climbing (MMHC) algorithm`
The skeleton of a Bayesian network as produced by MMHC

Description

The skeleton of a Bayesian network produced by MMHC. No orientations are involved.

Usage

```
mmhc.skel(dataset, max_k = 3, threshold = 0.05, test = "testIndFisher", rob = FALSE,
fast = FALSE, symmetry = TRUE, nc = 1, graph = FALSE)
```

Arguments

`dataset`	A matrix with the variables. The user must know if they are continuous or if they are categorical. data.frame or matrix are both supported, as the dataset is converted into a matrix.
`max_k`	The maximum conditioning set to use in the conditional indepedence test (see Details of SES or MMPC).
`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 "testIndFisher". This procedure allows for "testIndFisher", "testIndSPearman" for continuous variables and "gSquare" for categorical variables.
`rob`	A boolean variable which indicates whether (TRUE) or not (FALSE) to use a robust version of the statistical test if it is available. It takes more time than a non robust version but it is suggested in case of outliers. Default value is FALSE. This will only be taken into account if test is "testIndFisher".
`fast`	A bollean variable indicating a faster procedure to take place. By default this is set to FALSE. See details about this.
`symmetry`	In order for an edge to be added, a statistical relationship must have been found from both directions. If you want this symmetry correction to take place, leave this boolean variable to TRUE. If you set it to FALSE, then if a relationship between Y and X is detected but not between X and Y, the edge is still added.
`nc`	How many cores to use. This plays an important role if you have many variables, say thousands or so. You can try with nc = 1 and with nc = 4 for example to see the differences. If you have a multicore machine, this is a must option.
`graph`	Boolean that indicates whether or not to generate a plot with the graph. The package "RgraphViz" is required.

Details

The MMPC is run on every variable. The backward phase (see Tsamardinos et al., 2006) takes place automatically. After all variables have been used, the matrix is checked for inconsistencies and they are corrected.

A trick mentioned in that paper to make the procedure faster is the following. In the k-th variable, the algorithm checks how many previously scanned variables have an edge with the this variable and keeps them (it discards the other variables with no edge) along with the next (unscanned) variables.

This trick reduces time, but can lead to different results. For example, if the i-th variable is removed, the k-th node might not remove an edge between the j-th variable, simply because the i-th variable that could d-sepate them is missing.

The user is given this option via the argument "fast", which can be either TRUE or FALSE. Parallel computation is also available.

Value

A list including:

`runtime`	The run time of the algorithm. A numeric vector. The first element is the user time, the second element is the system time and the third element is the elapsed time.
`density`	The number of edges divided by the total possible number of edges, that is #edges / $n(n-1)/2$, where n is the number of variables.
`info`	Some summary statistics about the edges, minimum, maximum, mean, median number of edges.
`G`	The adjancency matrix. A value of 1 in G[i, j] appears in G[j, i] also, indicating that i and j have an edge between them.

Bear in mind that the values can be extracted with the $ symbol, i.e. this is an S3 class output.

Author(s)

Michail Tsagris

R implementation and documentation: Giorgos Athineou <athineou@csd.uoc.gr> and Michail Tsagris <mtsagris@csd.uoc.gr>

References

Tsamardinos, Brown and Aliferis (2006). The max-min hill-climbing Bayesian network structure learning algorithm. Machine learning, 65(1), 31-78.

See Also

`SES`, `MMPC`, `pc.skel`

Examples

```
# simulate a dataset with continuous data
dataset <- matrix(runif(1000 * 50, 1, 100), nrow = 1000 )
a1 <- mmhc.skel(dataset, max_k = 3, threshold = 0.05, test = "testIndFisher",
rob = FALSE, nc = 1)
a2 <- mmhc.skel(dataset, max_k = 3, threshold = 0.05, test = "testIndSpearman",
rob = FALSE, nc = 1)
a3 <- pc.con(dataset)
a4 <- pc.skel(dataset, R = 1)

a1$runtime
a2$runtime
a3$runtime
```

`Skeleton of the PC algorithm`

The skeleton of a Bayesian network produced by the PC algorithm

---

Description

The skeleton of a Bayesian network produced by the PC algorithm. No orientations are involved. The pc.con is a faster implementation for continuous datasets only. pc.skel is more general.

Usage

```
pc.skel(dataset, method = "pearson", alpha = 0.05, rob = FALSE, R = 1, graph = FALSE)

pc.con(dataset, method = "pearson", alpha = 0.05, graph = FALSE)
```

Arguments

`dataset`	A matrix with the variables. The user must know if they are continuous or if they are categorical. data.frame or matrix are both supported, as the dataset is converted into a matrix.
`method`	If you have continuous data, you can choose either "pearson", "spearman" or "distcor". The latter uses the distance correlation. If you have categorical data though, this must be "cat".
`alpha`	The significance level (suitable values in (0, 1)) for assessing the p-values. Default value is 0.05.
`rob`	A boolean variable which indicates whether (TRUE) or not (FALSE) to use a robust version of the statistical test if it is available. It takes more time than a non robust version but it is suggested in case of outliers. Default value is FALSE. This will only be taken into account if method is "pearson".
`R`	The number of permutations to be conducted. This is taken into consideration for the "pc.skel" only. The Pearson correlation coefficient is calculated and the p-value is assessed via permutations.

`graph`	Boolean that indicates whether or not to generate a plot with the graph. Package "RgraphViz" is required.

Details

The PC algorithm as proposed by Spirtes et al. (2000) is implemented. The variables must be either continuous or categorical, only. The skeleton of the PC algorithm is order independent, since we are using the third heuristic (Spirte et al., 2000, pg. 90). At every ste of the alogirithm use the pairs which are least statistically associated. The conditioning set consists of variables which are most statistically associated with each either of the pair of variables.

For example, for the pair (X, Y) there can be two coniditoning sets for example (Z1, Z2) and (W1, W2). All p-values and test statistics and degrees of freedom have been computed at the first step of the algorithm. Take the p-values between (Z1, Z2) and (X, Y) and between (Z1, Z2) and (X, Y). The conditioning set with the minimum p-value is used first. If the minimum p-values are the same, use the second lowest p-value. If the unlikely, but not impossible event of all p-values being the same, the test statistic divided by the degrees of freedom is used as a means of choosing which conditioning set is to be used first.

If two or more p-values are below the machine epsilon (.Machine$double.eps which is equal to 2.220446e-16), all of them are set to 0. To make the comparison or the ordering feasible we use the logarithm of the p-value. Hence, the logarithm of the p-values is always calculated and used.

In the case of the G^2 test of independence (for categorical data) we have incorporated a rule of thumb. I the number of samples is at least 5 times the number of the parameters to be estimated, the test is performed, otherwise, independence is not rejected (see Tsamardinos et al., 2006).

The pc.con is a faster implementation of the PC algorithm but for continuous data only, without the robust option, unlike pc.skel which is more general and even for the continuous datasets slower. pc.con accepts only "pearson" and "spearman" as correlations.

If there are missing values they are placed by their median in case of continuous data and by their mode (most frequent value) if they are categorical.

Value

A list including:

`stat`	The test statistics of the univariate associations.
`pvalue`	The logarithm of the p-values of the univariate associations.
`info`	Some summary statistics about the edges, minimum, maximum, mean, median number of edges.
`runtime`	The run time of the algorithm. A numeric vector. The first element is the user time, the second element is the system time and the third element is the elapsed time.
`kappa`	The maximum value of k, the maximum cardinality of the conditioning set at which the algorithm stopped.
`density`	The number of edges divided by the total possible number of edges, that is #edges / $n(n-1)/2$, where n is the number of variables.
`info`	Some summary statistics about the edges, minimum, maximum, mean, median number of edges.

`G`	The adjancency matrix. A value of 1 in G[i, j] appears in G[j, i] also, indicating that i and j have an edge between them.
`sepset`	A list with the separating sets for every value of k.
`title`	The name of the dataset.

Bear in mind that the values can be extracted with the $ symbol, i.e. this is an S3 class output.

Author(s)

Michail Tsagris

R implementation and documentation: Michail Tsagris <mtsagris@csd.uoc.gr>

References

Spirtes P., Glymour C. and Scheines R. (2001). Causation, Prediction, and Search. The MIT Press, Cambridge, MA, USA, 3nd edition. Szekely G.J. and Rizzo, M.L. (2014). Partial distance correlation with methods for dissimilarities. The Annals of Statistics, 42(6): 2382–2412.

Szekely G.J. and Rizzo M.L. (2013). Energy statistics: A class of statistics based on distances. Journal of Statistical Planning and Inference 143(8): 1249–1272.

See Also

`SES`, `MMPC`, `mmhc.skel`

Examples

```
# simulate a dataset with continuous data
dataset <- matrix( runif(1000 * 50, 1, 100), nrow = 1000 )
a <- mmhc.skel(dataset, max_k = 3, threshold = 0.05, test = "testIndFisher" )
b <- pc.skel( dataset, method = "pearson", alpha = 0.05 )
b2 <- pc.con( dataset, method = "pearson" )
a$runtime ##
b$runtime ##
b2$runtime ## check the diffrerences in the runtimes
```

`The max-min Markov blanket algorithm`

Max-min Markov blanket algorithm

Description

The MMMB algorithm follows a forward-backward filter approach for feature selection in order to provide a minimal, highly-predictive, feature subset of a high dimensional dataset. See also Details.

Usage

```
mmmb(target , dataset , max_k = 3 , threshold = 0.05 , test = "testIndFisher",
user_test = NULL, robust = FALSE, ncores = 1, hold = FALSE)
```

Arguments

`target`	The class variable. Provide either a string, an integer, a numeric value, a vector, a factor, an ordered factor or a Surv object. See also Details.
`dataset`	The dataset; provide either a data frame or a matrix (columns = variables, rows = samples). In either case, only two cases are avaialble, either all data are continuous, or categorical.
`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 the p-values. Default value is 0.05.
`test`	The conditional independence test to use. Default value is "testIndFisher". See also `link{CondIndTests}`.
`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.
`robust`	A boolean variable which indicates whether (TRUE) or not (FALSE) to use a robust version of the statistical test if it is available. It takes more time than a non robust version but it is suggested in case of outliers. Default value is FALSE.
`ncores`	How many cores to use. This plays an important role if you have tens of thousands of variables or really large sample sizes and tens of thousands of variables and a regression based test which requires numerical optimisation. In other cammmb it will not make a difference in the overall time (in fact it can be slower). The parallel computation is used in the first step of the algorithm, where univariate associations are examined, those take place in parallel. We have seen a reduction in time of 50% with 4 cores in comparison to 1 core. Note also, that the amount of reduction is not linear in the number of cores.
`hold`	After the MMPC, the backward (or symmetry correction) phase is implemented. This will remove any possibly falsely included variables in the parents and children set of the target variable and it will slow down the algorithm. If hold is TRUE, even if some variables are identified as falsely included, they will remain. If there are highly collinear (or statistically equivalent) variables, this phase tends to remove correctly identified variables, simply because it will identify a variable wich is highly collinear with the target variable. In this case, the hold should be TRUE. Can you know this in advnace? Well, maybe you can run the SES algorithm to get an idea, or be suspicious about it.

Details

The idea is to run the MMPC algorithm at first and identify the parents and children (PCt) of the target variable. As a second step, the MMPC algorithm is run on the discovered variables to return PCi. The parents of the children of the target are the spouses of the target. Every variable in PCi is checked to see if it is a spouse of the target. If yes, it is included in the Markov Blanket of the target, otherwise it is thrown. If the data are continous, the Fisher correlation test is used or the Spearman correlation (more robust). If the data are categorical, the G^2 test is used.

Value

The output of the algorithm is S3 object including:

`mb`	The Markov Blanket of the target variable. The parents and children of the target variable, along with the spouses of the target, which are the parents of the children of the target variable.
`runtime`	The run time of the algorithm. A numeric vector. The first element is the user time, the second element is the system time and the third element is the elapsed time.

Author(s)

Michail Tsagris

R implementation and documentation: Michail Tsagris <mtsagris@csd.uoc.gr>

References

Tsamardinos I., Aliferis C. F. and Statnikov, A. (2003). Time and sample efficient discovery of Markov blankets and direct causal relations. In Proceedings of the 9th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 673-678.

See Also

`CondIndTests`, `MMPC`, `SES`

Examples

```
set.seed(123)
require(hash)

#simulate a dataset with continuous data
dataset <- matrix( runif(1000 * 50, 1, 100), ncol = 50 )

#define a simulated class variable
target <- 3 * dataset[, 10] + 2 * dataset[, 50] + 3 * dataset[, 20] + rnorm(1000, 0, 5)

aa <- mmmb(target , dataset , max_k = 3 , threshold = 0.05, test= "testIndFisher", robust = FALSE,
ncores = 1, hold = FALSE)
ab <- SES(target, dataset, test="testIndFisher")
```

`Transformation of a DAG into an essential graph`
Transforms a DAG into an essential graph

Description

Transforms a DAG into an essential graph.

Usage

```
dag2eg(dag, type = NULL)
```

Arguments

`dag`	The graph matrix as produced from `pc.or` or any other algorithm which produces directed graphs. A DAG in general.
`type`	This can be either NULL or 1 or 2. type = 1 means that the matrix contains 0, 1, 2, 3 where G[i, j] = g[j, i] = 0, means there is no edge between nodes i and j, G[i, j] = g[j, i] = 1, there is an edge between nodes i and j and G[i, j] = 2 and G[j, i] = 3 means that there is an arrow from node i to node j. If type 2, the matrix contains 0 for no edge and 1 for a directed edge. In this case, G[i,j]=1 and G[j,i]=0 means that there is an arrow from node i to node j. If you are not sure of what you have, just leave it NULL, the function will check to which case your matrix belongs.

Details

The function is an R translation from an old matlab code.

Value

The matrix of the essential graph.

Author(s)

Ioannis Tsamardinos and Michail Tsagris

R implementation and documentation: Giorgos Athineou <athineou@csd.uoc.gr>, Ioannis Tsamardinos <tsamard@csd.uoc.gr> and and Michail Tsagris <mtsagris@csd.uoc.gr>

References

Chickering, D.M. (1995). A transformational characterization of equivalent Bayesian network structures. Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence, Montreal, Canada, 87-98.

See Also

`plotnetwork`, `nei`, `pc.or`

Examples

```
# simulate a dataset with continuous data
# simulate a dataset with continuous data
y = rdag(1000, 10, 0.3)
tru = y$G
eg = dag2eg(tru)
par( mfrow = c(1, 2) )
plotnetwork(tru)
plotnetwork(eg)
```

Transitive closure of an adjacency matrix

Returns the transitive closure of an adjacency matrix

Description

Returns the transitive closure of an adjacency matrix.

Usage

```
transitiveClosure(amat)
```

Arguments

`amat`	The adjacency matrix of a graph.

Details

A function that computes the transitive closure of a graph. The transitive closure C(G) of a graph is a graph which contains an edge between nodes u and v whenever there is a directed path from u to v (Skiena 1990, p. 203). http://mathworld.wolfram.com/TransitiveClosure.html

Value

`closure`	The transititve closure of the adjacency matrix representing a graph.

Author(s)

Anna Roumpelaki

R implementation and documentation: Anna Roumpelaki <anna.roumpelaki@gmail.com>

References

Skiena S. (1990). Implementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematica. Reading, MA: Addison-Wesley

Examples

```
# example adjacency matrix
# simulate a dataset with continuous data
dataset <- matrix( runif(1000 * 10, 1, 100), nrow = 1000 )
test <- pc.con( dataset, method = "pearson", alpha = 0.05 )$G
transitiveClosure(test)
```

`Undirected path(s) between two nodes` *Undirected path(s) between two nodes*

Description

Undirected path(s) between two nodes.

Usage

```
undir.path(G, y, x)
```

Arguments

`G`	An adjacency matrix where G[i,j] = G[j, i] = 1 means there is an edge between nodes i and j. If G[i, j] = G[j, i] = 0 there is no edge between them.
`y`	A numerical value indicating the first node, it has to be a number between 1 and the maximum number of variables.
`x`	A numerical value indicating the second node, it has to be a number between 1 and the maximum number of variables. The order of the nodes does not make a difference.

Details

The algorithm finds all the nodes between the two nodes. It finds all paths between the two chosen nodes.

Value

A vector with the two nodes and all nodes between them in the case of connecting nodes. Otherwise, a matrix with the neighbours of each node.

Author(s)

Michail Tsagris

R implementation and documentation: Michail Tsagris <mtsagris@csd.uoc.gr>

References

Tsamardinos, Brown and Aliferis (2006). The max-min hill-climbing Bayesian network structure learning algorithm. Machine learning, 65(1), 31-78.

See Also

`SES`, `MMPC`, `pc.skel`

Examples

```
# simulate a dataset with continuous data
set.seed(1234)
dataset <- matrix(runif(1000 * 20, 1, 100), nrow = 1000 )
G <- pc.con(dataset)$G
plotnetwork(G)
undir.path(G, 3, 4)
undir.path(G, 1, 3)
```

`Univariate regression based tests` *Univariate regression based tests*

Description

Univariate regression based tests.

Usage

```
univregs(target, dataset, test, wei = NULL, robust = FALSE, ncores = 1)
```

Arguments

`target`	The target (dependent) variable. It must be a numerical vector with integers.
`dataset`	The indendent variable(s). It can be a vector, a matrix or a dataframe with continuous only variables, a data frame with mixed or only categorical variables.
`test`	This argument should be one of the following strictly. **testIndFisher**, **testIndSpearman**, **gSquare**, **testIndBeta**, **testIndReg**, **testIndSpeedglm**, **testIndLogistic**, **testIndPois**, **testIndZip**, **censIndCR** or **censIndWR**. Note that you must give the name of the test, without "".
`wei`	A vector of weights to be used for weighted regression. The default value is NULL.
`robust`	A boolean variable which indicates whether (TRUE) or not (FALSE) to use a robust version of the statistical test if it is available. It takes more time than non robust version but it is suggested in case of outliers. Default value is FALSE as it is currently nor supported.
`ncores`	How many cores to use. This plays an important role if you have tens of thousands of variables or really large sample sizes and tens of thousands of variables and a regression based test which requires numerical optimisation. In other cases it will not make a difference in the overall time (in fact it can be slower). The parallel computation is used in the first step of the algorithm, where univariate associations are examined, those take place in parallel. We have seen a reduction in time of 50% with 4 cores in comparison to 1 core. Note also, that the amount of reduction is not linear in the number of cores.

Details

This function is more as a help function for SES and MMPC, but it can also be called directly by the user. In some, one should specify the regression model to use and the function will perform all simple regressions, i.e. all regression models between the target and each of the variables in the dataset.

Value

A list including:

`stat`	The value of the test statistic.
`pvalue`	The logged p-value of the test.
`flag`	A number, either 1 (test performed) or 0 (test not performed).

Author(s)

Michail Tsagris

R implementation and documentation: Michail Tsagris <mtsagris@csd.uoc.gr>

See Also

`SES`, `MMPC`, `CondIndTests`, `reg.fit`, `ridge.reg`

Examples

```
y <- rpois(100, 15)
x <- matrix( rnorm(100 * 20), ncol = 20)
a1 <- univregs(y, x, testIndPois)
```

`Zero inflated Poisson regression`

Zero inflated Poisson regression

Description

Zero inflated Poisson regression.

Usage

```
zip.mod(target, dataset, wei = NULL, xnew = NULL)
zip.reg(target, dataset, wei = NULL, lgy = NULL)
```

Arguments

`target`	The target (dependent) variable. It must be a numerical vector with integers.
`dataset`	The indendent variable(s). It can be a vector, a matrix or a dataframe with continuous only variables, a data frame with mixed or only categorical variables.
`wei`	A vector of weights to be used for weighted regression. The default value is NULL.
`xnew`	If you have new values for the predictor variables (dataset) whose target variable you want to predict insert them here. If you put the "dataset" or leave it NULL it will calculate the regression fitted values.
`lgy`	If you have already calculated the constant term of the ZIP regression plug it here. This is the sum of the logarithm of the factorial of the values.

Details

The zero inflated Poisson regression as suggested by Lambert (1992) is fitted. Unless you have a sufficient number of zeros, there is no reason to use this model. The "zip.reg" is an internal wrapper function and is used for speed up purposes. It is not to be called directly by the user unless they know what they are doing.

Value

A list including:

`iters`	The iterations required until convergence of the EM algorithm.
`be`	The estimated coefficients of the model.
`prop`	The estimated proportion of zeros.
`loglik`	The log-likelihood of the regression model.

Author(s)

Michail Tsagris

R implementation and documentation: Michail Tsagris <mtsagris@csd.uoc.gr>

References

Lambert D. (1992). Zero-inflated Poisson regression, with an application to defects in manufacturing. Technometrics, 34(1):1-14.

See Also

`testIndZIP`, `zip.regs`, `reg.fit`, `ridge.reg`

Examples

```
y <- rpois(100, 2)
x <- matrix( rnorm(100 * 2), ncol = 2)
a1 <- glm(y ~ x, poisson)
a2 <- zip.mod(y, x)
summary(a1)
logLik(a1)
a2  ## a ZIP is not really necessary
y[1:20] <- 0
a1 <- glm(y ~ x, poisson)
a2 <- zip.mod(y, x)
summary(a1)
logLik(a1)
a2  ## a ZIP is probably more necessary
```

Index