Package ‘MXM’

October 12, 2022

Type    Package
Title    Feature Selection (Including Multiple Solutions) and Bayesian Networks
Version  1.5.5
URL      http://mensxmachina.org
Date     2022-08-24
Author   Konstantina Biza [aut, cre],
         Ioannis Tsamardinos [aut, cph],
         Vincenzo Lagani [aut, cph],
         Giorgos Athineou [aut],
         Michail Tsagris [aut],
         Giorgos Borboudakis [ctb],
         Anna Roumpelaki [ctb]
Maintainer Konstantina Biza <kbiza@csd.uoc.gr>


License GPL-2
Depends R (>= 4.0)
Suggests markdown, R.rsp
VignetteBuilder knitr, R.rsp
Imports methods, stats, utils, survival, MASS, graphics, ordinal, nnet, quantreg, lme4, foreach, doParallel, parallel, relations, Rfast, visNetwork, energy, geepack, knitr, dplyr, bigmemory, coxme, Rfast2, Hmisc

NeedsCompilation no
Repository CRAN
Date/Publication 2022-08-25 08:52:40 UTC

R topics documented:

MXM-package .................................................. 5
Ancestors and descendants of a node in a directed graph .................................. 8
Backward phase of MMPC ...................................... 9
Backward selection regression ................................ 11
Backward selection regression for GLMM ................................ 12
Backward selection regression for GLMM using the eBIC ................................ 14
Backward selection regression using the eBIC .................................................. 15
Backward selection with generalised linear regression models ............................ 17
Bayesian Network construction using a hybrid of MMPC and PC ......................... 18
Beta regression .................................................. 21
BIC based forward selection ................................... 22
BIC based forward selection with generalised linear models .............................. 24
Bootstrap bias correction for the performance of the cross-validation procedure ...... 26
Calculation of the constant and slope for each subject over time ......................... 28
Certificate of exclusion from the selected variables set using SES or MMPC ........ 29
Check Markov equivalence of two DAGs ............................................... 30
Check whether a directed graph is acyclic ...................................................... 31
CondInditional independence tests ...................................................... 32
Conditional independence regression based tests ........................................... 40
Conditional independence test for binary, categorical or ordinal data ................. 43
Conditional independence test for case control data ....................................... 47
Conditional independence test for circular data ............................................ 49
Conditional independence test for longitudinal and clustered data using GEE ...... 51
Conditional independence test for longitudinal and clustered data using GLMM .... 55
Conditional independence test for proportions/percentages ............................... 58
Conditional independence test for the static-longitudinal scenario ...................... 60
Conditional independence tests counting the number of times a possible collider d- separates two nodes .............................................................. 63
Conditional independence tests for continuous univariate and multivariate data . . . . . . 65
Conditional independence tests for count data . . . . . . . . . . . . . . . . . . . . . . . 68
Conditional independence tests for left censored data . . . . . . . . . . . . . . . . . . 71
Conditional independence tests for positive data . . . . . . . . . . . . . . . . . . . . . 74
Conditional independence tests for success rates . . . . . . . . . . . . . . . . . . . . . 77
Conditional independence tests for survival data . . . . . . . . . . . . . . . . . . . . . 79
Conditional independence tests with and without permutation p-value . . . . . . . . . 83
Constraint based feature selection algorithms . . . . . . . . . . . . . . . . . . . . . . 85
Constraint based feature selection algorithms for longitudinal and clustered data . . . . 91
Constraint based feature selection algorithms for multiple datasets . . . . . . . . . . . 96
Correlation based conditional independence tests . . . . . . . . . . . . . . . . . . . . . 101
Cross-Validation for gOMP . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 104
Cross-validation for ridge regression . . . . . . . . . . . . . . . . . . . . . . . . . . . 107
Cross-Validation for SES and MMPC . . . . . . . . . . . . . . . . . . . . . . . . . . . 109
Cross-validation of the FBED with LMM . . . . . . . . . . . . . . . . . . . . . . . . . 114
Data simulation from a DAG . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 116
Drop all possible single terms from a model using the partial correlation . . . . . . . . 118
eBIC for many regression models . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 119
Effective sample size for G^2 test in BNs with case control data . . . . . . . . . . . . 121
Estimation of the percentage of Null p-values . . . . . . . . . . . . . . . . . . . . . . . 122
Fast MMPC . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 123
Fast MMPC for longitudinal and clustered data . . . . . . . . . . . . . . . . . . . . . . 126
Feature selection using SES and MMPC for classification with longitudinal data . . . . 130
Fit a mixture of beta distributions in p-values . . . . . . . . . . . . . . . . . . . . . . . 133
Forward Backward Early Dropping selection regression . . . . . . . . . . . . . . . . . . 134
Forward Backward Early Dropping selection regression for big data . . . . . . . . . . . 137
Forward Backward Early Dropping selection regression with GEE . . . . . . . . . . . . 139
Forward Backward Early Dropping selection regression with GLMM . . . . . . . . . . . 142
Forward selection regression . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 145
Forward selection with generalised linear regression models . . . . . . . . . . . . . . . 148
Forward selection with linear regression models . . . . . . . . . . . . . . . . . . . . . . 149
G-square conditional independence test for discrete data . . . . . . . . . . . . . . . . . 151
Generalised linear mixed models based on glmm SES and MMPC outputs . . . . . . . . 153
Generalised ordinal regression . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 156
Generate random folds for cross-validation . . . . . . . . . . . . . . . . . . . . . . . . . 157
Generic orthogonal matching pursuit (gOMP) . . . . . . . . . . . . . . . . . . . . . . . 158
Generic orthogonal matching pursuit (gOMP) for big data . . . . . . . . . . . . . . . . 161
Graph of unconditional associations . . . . . . . . . . . . . . . . . . . . . . . . . . . . 163
IAMB backward selection phase . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 164
IAMB variable selection . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 166
Incremental BIC values and final regression model of the FBED algorithm . . . . . . . . 168
Interactive plot of an (un)directed graph . . . . . . . . . . . . . . . . . . . . . . . . . 169
Lower limit of the confidence of an edge . . . . . . . . . . . . . . . . . . . . . . . . . . 170
mammpc.output-class . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 171
Many approximate simple logistic regressions . . . . . . . . . . . . . . . . . . . . . . . 172
Many simple beta regressions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 174
Many simple quantile regressions using logistic regressions . . . . . . . . . . . . . . . . 175
Many simple zero inflated Poisson regressions . . . . . . . . . . . . . . . . . . . . . . . 176
Many Wald based tests for logistic and Poisson regressions with continuous predictors . 178
Markov Blanket of a node in a directed graph .............................. 179
mases.output-class ............................................................... 180
MMPC solution paths for many combinations of hyper-parameters ... 181
MMPC.gee.output-class ......................................................... 184
MMPC.glmm.output-class ....................................................... 185
MMPCoutput-class .............................................................. 186
MXM-internal ................................................................. 187
Neighbours of nodes in an undirected graph .............................. 193
Network construction using the partial correlation based forward regression or FBED . 194
Orientation rules for the PC algorithm ...................................... 196
Partial correlation between two variables ................................. 198
Permutation based p-value for the Pearson correlation coefficient .. 199
Plot of longitudinal data ....................................................... 200
Probability residual of ordinal logistic regression ....................... 201
Read big data or a big.matrix object ............................ 202
Regression modeler ............................................................. 203
Regression models based on SES and MMPC outputs ................. 205
Regression models based on SES.timeclass and MMPC.timeclass outputs .......................... 207
Regression models fitting ..................................................... 208
Ridge regression ............................................................... 210
Ridge regression coefficients plot ......................................... 212
ROC and AUC ................................................................. 213
Search for triangles in an undirected graph ............................ 214
SES.gee.output-class ......................................................... 215
SES.glmm.output-class ......................................................... 216
SESoutput-class .............................................................. 217
Skeleton (local) around a node of the max-min hill-climbing (MMHC) algorithm .. 218
Skeleton of the max-min hill-climbing (MMHC) algorithm ............ 220
Skeleton of the PC algorithm .............................................. 223
Structural Hamming distance between two partially oriented DAGs . 227
Supervised PCA ............................................................... 229
Symmetric conditional independence test with clustered data ....... 230
Symmetric conditional independence test with mixed data ........ 232
The max-min Markov blanket algorithm ................................ 233
Topological sort of a DAG ............................................... 235
Total causal effect of a node on another node .......................... 236
Transformation of a DAG into an essential graph ...................... 238
Transitive closure of an adjacency matrix ............................... 239
Undirected path(s) between two nodes .................................. 240
Univariate regression based tests ......................................... 241
Utilities for the skeleton of a (Bayesian) Network ................. 245
Variable selection using the PC-simple algorithm .................... 247
Zero inflated Poisson and negative binomial regression ............. 248
This is an R package that currently implements feature selection methods for identifying minimal, statistically-equivalent and equally-predictive feature subsets. Additionally, the package includes two algorithms for constructing the skeleton of a Bayesian network.

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 correlation, 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 regression 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.

For more information the reader is addressed to


Details

Package: MXM
Type: Package
Version: 1.5.5
Date: 2022-08-24
License: GPL-2

Maintainer

Konstantina Biza <kbiza@csd.uoc.gr>.
Note

Acknowledgments: The research leading to these results has received funding from the European Research Council under the European Union’s Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreement n. 617393.

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. Their help has been very valuable. Dr Uwe Ligges and Prof Kurt Hornik from the CRAN team are greatly acknowledged for their assistance. Prof Achim Zeileis is greatly acknowledged for this help with the quasi Poisson and quasi binomial regression models. Christina Chatzipantsiou and Kleio Maria Verrou are acknowledged for their suggestions. Nikolaos Pandis from the University of Bern is acknowledged for his suggestion of the AFT (regression) models and for his suggestions. Michail is grateful to James Dalgleish from Columbia University who suggested that we mention, in various places, that most algorithms return the logarithm of the p-values and not the p-values. Stavros Lymeriadiis provided a very useful example where weights are used in a regression model; in surveys when stratified random sampling has been applied. Dr David Gomez Cabrero Lopez is also acknowledged. Margarita Rebolledo is acknowledged for spotting a bug. Zurab Khasidashvili from Intel Israel is acknowledged for spotting a bug in the function mnammb(). Teny Handhayani (PhD student at the University of York) spotted a bug in the conditional independence tests with mixed data and she is acknowledged for that. Dr. Kjell Jorner (Postdoctoral Fellow at the Department of Chemistry, University of Toronto) spotted a bug in two performance metrics of the bbc() function and he is acknowledged for that.

Disclaimer: Professor Tsamardinos is the creator of this package and Dr Lagani supervised Mr Athineou build it. Dr Tsagris is the current maintainer.

Author(s)

Ioannis Tsamardinos <tsamard@cs.d.uoc.gr>, Vincenzo Lagani <vlagani@cs.d.uoc.gr>, Giorgos Athineou <athineou@cs.d.uoc.gr>, Michail Tsagris <mtsagris@uoc.gr>, Giorgos Borboudakis <borboudakis@cs.d.uoc.gr>, Anna Roumelaki <anna.roumelaki@gmail.com>, Konstantina Biza <kbiza@cs.d.uoc.gr>.

References


Ancestors and descendants of a node in a directed graph


See Also

SES, MMPC, fbed.reg, gomp, pc.sel, 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>
### Description

Backward phase of MMPC.

### Usage

```r
mmpcbackphase(target, dataset, max_k = 3, threshold = 0.05, test = NULL, wei = NULL, R = 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).
- **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. Type the test without " ", e.g. type `testIndFisher`, Not "testIndFisher". Default value is NULL. See also `CondIndTests`.
- **wei**: A vector of weights to be used for weighted regression. The default value is NULL.

---

**Examples**

```r
# 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)
```
The number of permutations, set to 1 by default (no permutations based test). There is a trick to avoid doing all permutations. As soon as the number of times the permuted test statistic is more than the observed test statistic is more than 50 (if threshold = 0.05 and R = 999), the p-value has exceeded the significance level (threshold value) and hence the predictor variable is not significant. There is no need to continue do the extra permutations, as a decision has already been made.

Details

For each of the selected variables (dataset) the function performs conditional independence tests where the conditioning sets are formed from the other variables. All possible combinations are tried until the variable becomes non significant. The maximum size of the conditioning set is equal to max_k. This is called in the MMPC when the backward phase is requested.

Value

A list including:

- **met**: A numerical vector of size equal to the number of columns of the dataset.
- **counter**: The number of tests performed.
- **pvalues**: The maximum **logged p-value** for the association of each predictor variable.

Author(s)

Ioannis Tsamardinos, Michail Tsagris
R implementation and documentation: Michail Tsagris <mtsagris@uoc.gr>

References


See Also

MMPC, mmhc.skel, CondIndTests, cv.mmpc

Examples

```r
set.seed(123)

# simulate a dataset with continuous data
dataset <- matrix(runif(500 * 100, 1, 100), ncol = 100)

# define a simulated class variable
target <- 3 * dataset[, 10] + 2 * dataset[, 100] + 3 * dataset[, 20] + rnorm(500, 0, 5)

# MMPC algorithm
m1 <- MMPC(target, dataset, max_k = 3, threshold = 0.05, test="testIndFisher");
m2 <- MMPC(target, dataset, max_k = 3, threshold = 0.05, test="testIndFisher", backward = TRUE);
```
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)

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. For the gamma regression this must a vector with strictly positive numbers (no zeros allowed).
dataset The dataset; provide either a data frame or a matrix (columns = variables, rows = samples). In either case, only two cases are available, 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 "censIndER", "testIndMVreg", "testIndSpearman". 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. An example where weights are used is surveys when stratified sampling has occurred.
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.

Details

If the sample size is less than the number of variables a message 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.
Backward selection regression for GLMM

info A matrix with the non selected variables and their latest test statistics and logged p-values.

mat A matrix with the selected variables and their latest test statistics and logged p-values.

ci_test The conditional independence test used.

final The final regression model.

Author(s)
Michail Tsagris
R implementation and documentation: Michail Tsagris <mtsagris@uoc.gr>

See Also
glm.fsreg, lm.fsreg, bic.fsreg, bic.glm.fsreg, CondIndTests, MMPC, SES

Examples
set.seed(123)
dataset <- matrix( runif(200 * 10, 1, 100), ncol = 10 )
target <- rnorm(200)
a <- bs.reg(target, dataset, threshold = 0.05, test = "testIndRQ")
b <- bs.reg(target, dataset, threshold = 0.05, test = "testIndGLMMReg")
b2 <- bs.reg(target, dataset, threshold = 0.05, test = "testIndFisher")

Backward selection regression for GLMM

Backward selection regression for GLMM

Description
Backward selection regression for GLMM

Usage
glmm.bsreg(target, dataset, id, threshold = 0.05, wei = NULL, test = "testIndGLMMReg")

Arguments
target The class variable. This can be a numerical vector with continuous data, binary or discrete valued data. It can also be a factor variable with two levels only.
dataset The dataset; provide a numerical a matrix (columns = variables, rows = samples).
id This is a numerical vector of the same size as target denoting the groups or the subjects.
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.

test

This is for the type of regression to be used, "testIndGLMMReg", for Gaussian regression, "testIndGLMMLogistic" for logistic regression or "testIndGLMM-Pois" for Poisson regression.

Details

If the sample size is less than the number of variables a message 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 logged p-values.

mat

A matrix with the selected variables and their latest statistics and logged p-values.

final

The final regression model.

Author(s)

Michail Tsagris

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

References


See Also

fbed.glmm.reg, ebic.glmm.bsreg, MMPC.glmm

Examples

```r
## Not run:
require(lme4)
data(sleepstudy)
reaction <- sleepstudy$Reaction
days <- sleepstudy$Days
subject <- sleepstudy$Subject
x <- matrix(rnorm(180 * 200), ncol = 200) ## unrelated predictor variables
```
m1 <- glmm.bsreg(Reaction, x, subject)
m2 <- MMPC.glmm(target = reaction, group = subject, dataset = x)

## End(Not run)

Backward selection regression for GLMM using the eBIC

Description
Backward selection regression for GLMM using the eBIC

Usage
ebic.glmm.bsreg(target, dataset, id, wei = NULL, gam = NULL, test = "testIndGLMMReg")

Arguments
- **target**: The class variable. This can be a numerical vector with continuous data, binary or discrete valued data. It can also be a factor variable with two levels only.
- **dataset**: The dataset; provide a numerical a matrix (columns = variables, rows = samples).
- **id**: This is a numerical vector of the same size as target denoting the groups or the subjects.
- **wei**: A vector of weights to be used for weighted regression. The default value is NULL. An example where weights are used is surveys when stratified sampling has occurred.
- **gam**: In case the method is chosen to be "eBIC" one can also specify the gamma parameter. The default value is "NULL", so that the value is automatically calculated.
- **test**: This is for the type of regression to be used, "testIndGLMMReg", for Gaussian regression, "testIndGLMMLogistic" for logistic regression or "testIndGLMM-Pois" for Poisson regression.

Details
The algorithm is a variation of the usual forward selection. At every step, the most significant variable enters the selected variables set. In addition, only the significant variables stay and are further examined. The non significant ones are dropped. This goes until no variable can enter the set. The user has the option to redo this step 1 or more times (the argument K). In the end, a backward selection is performed to remove falsely selected variables.
Value

A list including:

- runtime: The runtime required.
- info: A matrix with the number of variables and the number of tests performed (or models fitted) at each round (value of K).
- mat: A matrix with the selected variables and their eBIC.

Author(s)

Michail Tsagris

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

References


See Also

fbed.glmm.reg, glmm.bsreg, MMPC.glmm

Examples

```r
## Not run:
require(lme4)
data(sleepstudy)
reaction <- sleepstudy$Reaction
days <- sleepstudy$Days
subject <- sleepstudy$Subject
x <- matrix(rnorm(180 * 20), ncol = 20) ## unrelated predictor variables
m1 <- ebic.glmm.bsreg(reaction, x, id = subject)
m2 <- MMPC.glmm(reaction, group = subject, dataset = x)
## End(Not run)
```

Description

Backward selection regression using the eBIC
Backward selection regression using the eBIC

Usage

```
ebic.bsreg(target, dataset, test = NULL, wei = NULL, gam = NULL)
```

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 = samples).
test
The available tests: "testIndReg", "testIndPois", "testIndNB", "testIndLogistic", "testIndMMReg", "testIndBinom", "censIndCR", "censIndWR", "testIndBeta", "testIndZIP", "testIndGamma", "testIndNormLog" and "testIndTobit".
wei
A vector of weights to be used for weighted regression. The default value is NULL. It is not suggested when testIndMMReg is used. An example where weights are used is surveys when stratified sampling has occured.
gam
In case the method is chosen to be "eBIC" one can also specify the gamma parameter. The default value is "NULL", so that the value is automatically calculated.

details
The algorithm is a variation of the usual forward selection. At every step, the most significant variable enters the selected variables set. In addition, only the significant variables stay and are further examined. The non significant ones are dropped. This goes until no variable can enter the set. The user has the option to redo this step 1 or more times (the argument K). In the end, a backward selection is performed to remove falsely selected variables.

Value

A list including:

```
runtime
info
mat
back.rem
back.n.tests
```

runtime
The runtime required.
info
A matrix with the number of variables and the number of tests performed (or models fitted) at each round (value of K).
mat
A matrix with the selected variables and their eBIC.
back.rem
The variables removed in the backward phase.
back.n.tests
The number of models fitted in the backward phase.

Author(s)

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

References

Backward selection with generalised linear regression models

See Also

fs.reg, bic.fsreg, MMPC

Examples

dataset <- matrix( runif(100 * 15, 1, 100), ncol = 15 )
target <- rt(100, 10)
a1 <- ebic.bsreg(target, dataset, test = "testIndReg")
target <- rpois(100, 10)
a2 <- ebic.bsreg(target, dataset, test = "testIndPois")

Description

Variable selection in generalised linear regression models with backward selection

Usage

glm.bsreg(target, dataset, threshold = 0.05, wei = NULL, test = NULL)
glm.bsreg2(target, dataset, threshold = 0.05, wei = NULL, test = NULL)

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 number 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 available, 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. An example where weights are used is surveys when stratified sampling has occurred.
test For "glm.bsreg" this can be "testIndLogistic", "testIndPois", "testIndBinom", "testIndReg" or "testIndMMReg". For "glm.bsreg2" this can be "testIndGamma", "testIndNormLog", "testIndQPois" or "testIndQBinom".

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 message will appear and no backward regression is performed.
Bayesian Network construction using a hybrid of MMPC and PC

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 **logged p-values**.

- **mat**
  A matrix with the selected variables and their latest test statistic and **logged 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@uoc.gr>

See Also
- fs.reg, lm.fsreg, bic.fsreg, bic.glm.fsreg, CondIndTests, MMPC, SES

Examples

```r
set.seed(123)

# simulate a dataset with continuous data
dataset <- matrix( runif(200 * 10, 1, 100), ncol = 10 )

# define a simulated class variable
target <- rpois(200, 10)
a <- glm.bsreg(target, dataset, threshold = 0.05)

target <- rbinom(200, 1, 0.6)
b <- glm.bsreg(target, dataset, threshold = 0.05)

target <- rgamma(200, 1, 2)
b1 <- glm.bsreg2(target, dataset, threshold = 0.05, test = "testIndGamma")
b2 <- glm.bsreg2(target, dataset, threshold = 0.05, test = "testIndNormLog")
```

Bayesian Network construction using a hybrid of MMPC and PC
Usage

```r
mmpc.or(x, max_k = 5, threshold = 0.01, test = "testIndFisher", backward = TRUE, symmetry = TRUE, ini.pvalue = NULL)
```

Arguments

- **x**: A matrix with the variables. The user must know if they are continuous or if they are categorical. If you have a matrix with categorical data, i.e., 0, 1, 2, 3 where each number indicates a category, the minimum number for each variable must be 0. data.frame is also supported, as the dataset in this case is converted into a matrix.

- **max_k**: The maximum conditioning set to use in the conditional independence 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.

- **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. It calls the `link{mmpcbackphase}` for this purpose. For perm.ses and wald.ses this is not yet applicable.

- **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.

- **ini.pvalue**: This is a list with the matrix of the univariate p-values. If you want to run mmhc.skel again, the univariate associations need not be calculated again.

Details

The MMPC is run on every variable. The backward phase (see Tsamardinos et al., 2006) can then take place. After all variables have been used, the matrix is checked for inconsistencies and they are corrected if you want. The "symmetry" argument. Do you want the edge to stay if it was discovered from both variables when they were considered as responses?

Value

A list including:

- **ini.pvalue**: A matrix with the p-values of all pairwise univariate associations.

- **kapa**: The maximum number of conditioning variables ever observed.

- **ntests**: The number of tests MMPC (or SES) performed at each variable.

- **info**: Some summary statistics about the edges, minimum, maximum, mean, median number of edges.
Bayesian Network construction using a hybrid of MMPC and PC

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.

runtime
   The run time of the skeleton phase 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.

runtime.or
   The run time of the PC orientation rules. 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.

Gini
   The adjacency 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.

G
   The final adjacency 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.

sepset
   A list with the separating sets for every value of k.

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@uoc.gr>

References


Cambridge, MA, USA, 3rd edition.

See Also

pc.skel, pc.or, corfs.network

Examples

```r
y <- rdag2(500, p = 20, nei = 3)
ind <- sample(1:20, 20)
x <- y$x[, ind]
a1 <- mmpc.or(x, max_k = 3, threshold = 0.01, test = "testIndFisher")
b <- pc.skel(x, alpha = 0.01)
a2 <- pc.or(b)
```
Beta regression

Description

Beta regression.

Usage

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

Arguments

- **target**: The target (dependent) variable. It must be a numerical vector with proportions, excluding 0s and 1s.
- **dataset**: The independent 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. If this is NULL, a beta distribution is fitted, no covariates are present.
- **wei**: A vector of weights to be used for weighted regression. The default value is NULL. An example where weights are used is surveys when stratified sampling has occurred.
- **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.

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:

- **be**: The estimated coefficients of the model.
- **phi**: The estimated precision parameter.
- **loglik**: The log-likelihood of the regression model.
- **est**: The estimated values if xnew is not NULL.

Author(s)

Michail Tsagris

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

References

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, 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, "testIndGamma" for gamma regression, "testIndNormLog" for linear regression with the log-link (non negative data), "testIndTobit" for Tobit regression. 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. An example where weights are used is surveys when stratified sampling has occurred.

- **tol**: The difference between two successive values of the stopping rule. By default this is is set to 2. If for example, the BIC difference between two successive models is less than 2, the process stops and the last variable, even though significant does not enter the model.

Examples

```r
y <- rbeta(300, 3, 5)
x <- matrix(rnorm(300 * 2), ncol = 2)
a1 <- beta.mod(y, x)
w <- runif(300)
a2 <- beta.mod(y, x, w)
```
\textit{BIC based forward selection}

\begin{itemize}
  \item \texttt{ncores} \quad 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 camma bb 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.
\end{itemize}

\textbf{Details}

If the current \textquote{test} argument is defined as NULL or \textquote{auto} and the \texttt{user_test} argument is NULL then the algorithm automatically selects the best test based on the type of the data. Particularly:

\begin{itemize}
  \item 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.
  \item if target is a ordered factor, the ordinal regression is used.
  \item if target is a numerical vector or a matrix with at least two columns (multivariate) linear regression is used.
  \item 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.
  \item if target is a \texttt{Surv} object, the Survival conditional independence test (Cox regression) is used.
\end{itemize}

\textbf{Value}

The output of the algorithm is S3 object including:

\begin{itemize}
  \item \texttt{runtime} \quad 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.
  \item \texttt{mat} \quad A matrix with the variables and their latest test statistics and p-values.
  \item \texttt{info} \quad A matrix with the selected variables, and the BIC of the model with that and all the previous variables.
  \item \texttt{ci_test} \quad The conditional independence test used.
  \item \texttt{final} \quad The final regression model.
\end{itemize}

\textbf{Author(s)}

Michail Tsagris

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

\textbf{References}

BIC based forward selection with generalised linear models

See Also

\texttt{glm.fsreg}, \texttt{lm.fsreg}, \texttt{bic glm.fsreg}, \texttt{CondIndTests}, \texttt{MMPC}, \texttt{SES}

Examples

```r
set.seed(123)
dataset <- matrix( rrunif(200 * 20, 1, 100), ncol = 20 )
target <- 3 * dataset[, 10] + 2 * dataset[, 15] + 3 * dataset[, 20] + rnorm(200, 0, 5)

a1 <- bic.fsreg(target, dataset, tol = 4, ncores = 1, test = "testIndReg")
a3 <- MMPC(target, dataset, ncores = 1)
target <- round(target)
b1 <- bic.fsreg(target, dataset, tol = 2, ncores = 1, test = "testIndReg")
```

---

BIC based forward selection with generalised linear models

\textit{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

\begin{verbatim}
bic glm.fsreg( target, dataset, wei = NULL, tol = 0, ncores = 1 )
bic mm.fsreg( target, dataset, wei = NULL, tol = 0, ncores = 1 )
bic gammafsreg( target, dataset, wei = NULL, tol = 0, ncores = 1 )
bic normlog.fsreg( target, dataset, wei = NULL, tol = 0, ncores = 1 )
\end{verbatim}

Arguments

\begin{itemize}
\item \texttt{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.
\item \texttt{dataset} The dataset; provide either a data frame or a matrix (columns = variables, rows = samples). These can be continuous and or categorical.
\item \texttt{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.
\item \texttt{tol} The difference between two successive values of BIC. By default this is is set to 2. If for example, the BIC difference between two successive models is less than 2, the process stops and the last variable, even though significant does not enter the model.
\end{itemize}
BIC based forward selection with generalised linear models

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:

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 logged p-values.

info
A matrix with the selected variables, and the BIC of the model with that and all the previous variables.

ci_test
The conditional independence test used.

final
The final regression model.

Author(s)

Michail Tsagris

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

See Also

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

Examples

set.seed(123)
dataset <- matrix( runif(200 * 20, 1, 100), ncol = 20 )
target <- 3 * dataset[, 10] + 2 * dataset[, 15] + 3 * dataset[, 20] + rnorm(200, 0, 5)
a1 <- bic.glm.fsreg(target, dataset, tol = 2, ncores = 1)
a2 <- bic.glm.fsreg( round(target), dataset, tol = 2, ncores = 1)
y <- target ; me <- median(target) ; y[ y < me ] <- 0 ; y[ y >= me ] <- 1
a3 <- bic.glm.fsreg( y, dataset, tol = 2, ncores = 1)
Bootstrap bias correction for the performance of the cross-validation procedure

Description

Bootstrap bias correction for the performance of the cross-validation procedure.

Usage

```
bbc(predictions, target, metric = "auc.mxm", conf = 0.95, B = 1000)
```

Arguments

- `predictions`: A matrix with the predicted values.
- `target`: A vector with the target variable, survival object, factor (ordered or unordered) or a numerical vector.
- `metric`: The possible values are:
  - a) Binary target: "auc.mxm" (area under the curve), "fscore.mxm" (F-score), "prec.mxm" (precision), "euclid_sens.spec.mxm" (Euclidean distance of sensitivity and specificity), "spec.mxm" (specificity), "sens.mxm" (sensitivity), "acc.mxm" (accuracy, proportion of correct classification).
  - b) Multinomial target: "acc_multinom.mxm" (accuracy, proportion of correct classification).
  - c) Ordinal target: "ord_mae.mxm" (mean absolute error).
  - d) Continuous target: "mae.mxm" (MAE with continuous target), "mse.mxm" (mean squared error), "pve.mxm" (percentage of variance explained).
  - e) Survival target "ci.mxm" (concordance index for Cox regression), "ciwr.mxm" (concordance index for Weibull regression).
  - g) Count target "poisdev.mxm".
  - h) Binomial target "binomdev.mxm" (deviance of binomial regression).

The "nbdev.mxm" (negative binomial deviance) is missing. For more information on these see `cv.ses`. boldNote that they come with "".

- `conf`: A number between 0 and 1, the confidence level.
- `B`: The number of bootstrap replicates. The default number is 1000.

Details

Upon completion of the cross-validation, the predicted values produced by all predictive models across all folds is collected in a matrix $P$ of dimensions $n \times M$, where $n$ is the number of samples and $M$ the number of trained models or configurations. Sampled with replacement a fraction of rows (predictions) from $P$ are denoted as the in-sample values. On average, the newly created set will be comprised by 63.2% of the original individuals (The probability of sampling, with replacement, a sample of $n$ numbers from a set of $n$ numbers is $1 - \left(1 - \frac{1}{n}\right)^n \approx 1 - \frac{1}{e} = 0.632$), whereas
the rest 36.8% will be random copies of them. The non re-sampled rows are denoted as out-of-sample values. The performance of each model in the in-sample rows is calculated and the model (or configuration) with the optimal performance is selected, followed by the calculation of performance in the out-of-sample values. This process is repeated B times and the average performance is returned.

Note, that the only computational overhead is with the repetitive re-sampling and calculation of the predictive performance, i.e. no model is fitted nor trained. The final estimated performance usually underestimates the true performance, but this negative bias is smaller than the optimistic uncorrected performance.

Note, that all metrics are for maximization. For this reason "mse.mxm", "mae.mxm", "ord_mae.mxm", "poisdev.mxm", "binomdev.mxm" are multiplied by -1.

Value

A list including:

out.perf: The B out sampled performances. Their mean is the "bbc.perf" given above.

bbc.perf: The bootstrap bias corrected performance of the chosen algorithm, model or configuration.

ci: The (1 - conf)% confidence interval of the BBC performance. It is based on the empirical or percentile method for bootstrap samples. The lower and upper 2.5% of the "out.perf".

Author(s)

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

References


See Also

cv.ses, cv.gomp

Examples

predictions <- matrix(rbinom(200 * 100, 1, 0.7), ncol = 100)
target <- rbinom(200, 1, 0.5)
bbc(predictions, target, metric = "auc.mxm")
Calculation of the constant and slope for each subject over time

Description

Calculation of the constant and slope for each subject over time.

Usage

group.mvbetas(x, id, reps)

Arguments

x
  The dataset; provide a numerical a matrix. This should contain longitudinal data. Each variable is a column, and rows are longitudinal data.

id
  This is a numerical vector denoting the subjects. Its length must be equal to the number of rows of the x matrix.

reps
  This is a numerical vector with the time points. Its length must be equal to the number of rows of the x matrix.

Details

This function is used internally in SES and MMPC and does calculations required by the first step of the Static-Longitudinal scenario of Tsagris, Lagani and Tsamardinos (2018). The measurements of each subject are regressed against time. So, for each subject, we get the constant and intercept over time and this is repeated for every feature.

Value

A matrix. The first r ( = length(unique(id) ), the number of subjects ) rows contain the constants and the other r rows contain the slopes.

Author(s)

Michail Tsagris

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

References


Certificate of exclusion from the selected variables set using SES or MMPC

See Also

fbed.gee.reg, glmm.bsreg, MMPC.glmm

Examples

## assume these are longitudinal data, each column is a variable (or feature)
x <- matrix( rnorm(100 * 30), ncol = 30 )
id <- rep(1:20, each = 5)  ## 20 subjects
reps <- rep( seq(4, 12, by = 2), 20)  ## 5 time points for each subject
a <- group.mvbetas(x, id, reps)
dim(a)  ## 5 100
## these are the regression coefficients of the first subject's values on the
## reps (which is assumed to be time in this example)
a[c(1, 21), 1]
coef( lm( x[id == 1, 1] ~ reps[1:5] ) )

Certificate of exclusion from the selected variables set using SES or MMPC

Description

Information on why one or more variables were not selected.

Usage

certificate.of.exclusion(xIndex, sesObject = NULL, mmpcObject = NULL)
certificate.of.exclusion2(xIndex, mmpc2object)

Arguments

xIndex A numerical vector with the indices of the predictor variables.

sesObject If you ran SES, wald.ses or perm.ses, give the whole SES object here, otherwise
leave it NULL.

mmpcObject If you ran MMPC, wald.mmpc or prm.mmpc, give the whole MMPC object
here, otherwise leave it NULL.

mmpc2object If you ran mmpc2, give the whole MMPC object here.

Value

A list with the conditioning variables (if any), the test statistic and the logarithm of the p-value. In
case a variable has been selected a message appears.

Author(s)

Michail Tsagris

R implementation and documentation: Michail Tsagris <mtsagris@uoc.gr>
Check Markov equivalence of two DAGs

See Also

MMPC

Examples

```r
set.seed(123)
# simulate a dataset with continuous data
dataset <- matrix(runif(100 * 100, 1, 100), ncol = 100)
#define a simulated class variable
target <- 3 * dataset[, 10] + 2 * dataset[, 100] + 3 * dataset[, 20] + rnorm(100, 0, 5)
#define some simulated equivalences
dataset[, 15] <- dataset[, 10] + rnorm(100, 0, 2)
dataset[, 100] <- dataset[, 100] + rnorm(100, 0, 2)
dataset[, 20] <- dataset[, 100] + rnorm(100, 0, 2)
# run the SES algorithm
mod1 <- SES(target, dataset, max_k = 5, threshold = 0.05, test = "testIndFisher",
hash = TRUE, hashObject = NULL);
mod2 <- MMPC(target, dataset, max_k = 5, threshold = 0.05, test = "testIndFisher",
hash = TRUE, hashObject = NULL);
certificate.of.exclusion(c(10, 15, 30, 45, 20), mod1)
certificate.of.exclusion(c(10, 15, 30, 45, 20), NULL, mod2)
```

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 (regardless of the mark, arrowhead, tail or nothing) and b) they have the same unshielded colliders.
Check whether a directed graph is acyclic

Value

A list including:

- **apofasi**: A boolean variable, TRUE or FALSE.
- **mes**: A message specifying the result, the dimensions of the adjacency matrices do not match for example, or the number of adjacencies 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@uoc.gr>

References


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)
```

Description

Check whether a directed graph is acyclic.

Usage

```
is.dag(dag)
```
Arguments

dag  A square matrix representing a directed graph which contains either 0, 1 or 0, 2, and 3. In the first case where G[i, j] = 1, means there is an arrow from node i to node j. In the second case G[i, j] = 2 and G[j, i] = 3 means that there is an arrow from node i to node j, where the 2 indicates the arrowhead and the 3 indicates the arrowtail.

Details

The topological sort is performed. If it cannot be performed, NAs are returned. Hence, the functions checks for NAs.

Value

A logical value, TRUE if the matrix represents a DAG and FALSE otherwise.

Author(s)

Michail Tsagris

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

References


See Also

topological_sort, dag2eg, pc.or

Examples

```r
# simulate a dataset with continuous data
G <- rdag(100, 20, 0.3)$G
is.dag(G)  ## TRUE
```

---

**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 which are available for SES and MMPC are summarized in the below table.

<table>
<thead>
<tr>
<th>Target variable</th>
<th>Predictor variables</th>
<th>Available tests</th>
<th>Short explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Continuous</td>
<td>Continuous</td>
<td>testIndFisher</td>
<td>Partial correlation</td>
</tr>
<tr>
<td>Continuous</td>
<td>Continuous</td>
<td>testIndMMFisher</td>
<td>Robust partial correlation</td>
</tr>
<tr>
<td>Continuous</td>
<td>Continuous</td>
<td>testIndSpearman</td>
<td>Partial correlation</td>
</tr>
<tr>
<td>Continuous</td>
<td>Mixed</td>
<td>testIndMMReg</td>
<td>MM regression</td>
</tr>
<tr>
<td>Proportions</td>
<td>Continuous</td>
<td>testIndFisher</td>
<td>Partial correlation</td>
</tr>
<tr>
<td>Proportions</td>
<td>Continuous</td>
<td>testIndMMFisher</td>
<td>Robust partial correlation</td>
</tr>
<tr>
<td>Proportions</td>
<td>Continuous</td>
<td>testIndSpearman</td>
<td>Partial correlation</td>
</tr>
<tr>
<td>Proportions</td>
<td>Mixed</td>
<td>testIndReg</td>
<td>Linear regression</td>
</tr>
<tr>
<td>Proportions</td>
<td>Mixed</td>
<td>testIndMMReg</td>
<td>MM regression</td>
</tr>
<tr>
<td>Proportions</td>
<td>Mixed</td>
<td>testIndRQ</td>
<td>Median regression</td>
</tr>
<tr>
<td>Proportions</td>
<td>Mixed</td>
<td>testIndBeta</td>
<td>Beta regression</td>
</tr>
<tr>
<td>Strictly positive</td>
<td>Mixed</td>
<td>testIndQbinom</td>
<td>Quasi binomial regression</td>
</tr>
<tr>
<td>Strictly positive</td>
<td>Mixed</td>
<td>testIndIGreg</td>
<td>Inverse Gaussian regression</td>
</tr>
<tr>
<td>Strictly positive</td>
<td>Mixed</td>
<td>testIndGamma</td>
<td>Gamma regression</td>
</tr>
<tr>
<td>Non negative</td>
<td>Mixed</td>
<td>testIndNormLog</td>
<td>Gaussian regression with log link</td>
</tr>
<tr>
<td>Strictly Positive</td>
<td>Mixed</td>
<td>censIndWR</td>
<td>Weibull regression</td>
</tr>
<tr>
<td>Strictly Positive</td>
<td>Mixed</td>
<td>censIndER</td>
<td>Exponential regression</td>
</tr>
<tr>
<td>Strictly Positive</td>
<td>Mixed</td>
<td>censIndLLR</td>
<td>Log-logistic regression</td>
</tr>
<tr>
<td>Successes &amp; totals</td>
<td>Mixed</td>
<td>testIndBinom</td>
<td>Binomial regression</td>
</tr>
<tr>
<td>Discrete</td>
<td>Mixed</td>
<td>testIndPois</td>
<td>Poisson regression</td>
</tr>
<tr>
<td>Discrete</td>
<td>Mixed</td>
<td>testIndZIP</td>
<td>Zero Inflated</td>
</tr>
<tr>
<td>Discrete</td>
<td>Mixed</td>
<td>testIndNB</td>
<td>Negative binomial regression</td>
</tr>
<tr>
<td>Discrete</td>
<td>Mixed</td>
<td>testIndQPoission</td>
<td>Quasi Poisson regression</td>
</tr>
<tr>
<td>Factor with two</td>
<td>Mixed</td>
<td>testIndLogistic</td>
<td>Binary logistic regression</td>
</tr>
<tr>
<td>levels or binary</td>
<td>Mixed</td>
<td>testIndQBinom</td>
<td>Quasi binomial regression</td>
</tr>
<tr>
<td>Factor with two</td>
<td>Mixed</td>
<td>testIndMultinom</td>
<td>Multinomial logistic regression</td>
</tr>
<tr>
<td>levels or binary</td>
<td>Mixed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Factor with more</td>
<td>Mixed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>than two levels</td>
<td>Mixed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(unordered)</td>
<td>Mixed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Factor with more</td>
<td>Mixed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>than two levels</td>
<td>Mixed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(ordered)</td>
<td>testIndOrdinal</td>
<td>Ordinal logistic regression</td>
<td></td>
</tr>
</tbody>
</table>
## Conditional independence tests

<table>
<thead>
<tr>
<th>Type of data</th>
<th>Test Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Categorical-Categorical</td>
<td>gSquare</td>
<td>G-squared test of independence</td>
</tr>
<tr>
<td>Categorical-Categorical</td>
<td>testIndMultinom</td>
<td>Multinomial logistic regression</td>
</tr>
<tr>
<td>Categorical-Categorical</td>
<td>testIndOrdinal</td>
<td>Ordinal logistic regression</td>
</tr>
<tr>
<td>Survival-Mixed</td>
<td>censIndCR</td>
<td>Cox regression</td>
</tr>
<tr>
<td>Survival-Mixed</td>
<td>censIndWR</td>
<td>Weibull regression</td>
</tr>
<tr>
<td>Survival-Mixed</td>
<td>censIndER</td>
<td>Exponential regression</td>
</tr>
<tr>
<td>Survival-Mixed</td>
<td>censIndGLLR</td>
<td>Log-logistic regression</td>
</tr>
<tr>
<td>Left censored-Mixed</td>
<td>testIndTobit</td>
<td>Tobit regression</td>
</tr>
<tr>
<td>Case-control-Mixed</td>
<td>testIndClogit</td>
<td>Conditional logistic regression</td>
</tr>
<tr>
<td>Multivariate continuous (no zeros)</td>
<td>testIndMVreg</td>
<td>Multivariate linear regression</td>
</tr>
<tr>
<td>Compositional data</td>
<td>testIndMVreg</td>
<td>Multivariate linear regression</td>
</tr>
<tr>
<td>Longitudinal/clustered</td>
<td>testIndGLMMReg</td>
<td>Linear mixed models</td>
</tr>
<tr>
<td>Clustered</td>
<td>testIndLMM</td>
<td>Fast linear mixed models</td>
</tr>
<tr>
<td>Binary longitudinal and clustered</td>
<td>testIndGLMMLogistic</td>
<td>Logistic mixed regression</td>
</tr>
<tr>
<td>Count longitudinal and clustered</td>
<td>testIndGLMMPois</td>
<td>Poisson mixed regression</td>
</tr>
<tr>
<td>Positive longitudinal and clustered</td>
<td>testIndGLMMNormLog</td>
<td>GLMM with Gaussian regression and log link</td>
</tr>
<tr>
<td>Non negative longitudinal and clustered</td>
<td>testIndGLMMGamma</td>
<td>GLMM with Gamma regression and log link</td>
</tr>
<tr>
<td>Longitudinal/clustered</td>
<td>testIndGEEReg</td>
<td>GEE with Gaussian regression</td>
</tr>
<tr>
<td>Binary longitudinal and clustered</td>
<td>testIndGEELogistic</td>
<td>GEE with logistic regression</td>
</tr>
<tr>
<td>Count longitudinal and clustered</td>
<td>testIndGEEPois</td>
<td>GEE with Poisson regression</td>
</tr>
<tr>
<td>Positive longitudinal and clustered</td>
<td>testIndGEENormLog</td>
<td>GEE with Gaussian regression and log link</td>
</tr>
<tr>
<td>Non negative longitudinal and clustered</td>
<td>testIndGEEGamma</td>
<td>GEE with Gamma regression and log link</td>
</tr>
<tr>
<td>Clustered survival</td>
<td>testIndGLMCR</td>
<td>Mixed effects Cox regression</td>
</tr>
<tr>
<td>Circular</td>
<td>testIndSPML</td>
<td>Circular-linear regression</td>
</tr>
</tbody>
</table>

### Details

These tests can be called by SES, MMPC, wald.mmpe or individually by the user. In all regression cases, there is an option for weights.

### Log-likelihood ratio 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 applied. 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),
gamma regression (testIndGamma) or Gaussian regression with log link (testIndNormLog).

8. testIndGamma. This is an alternative to testIndIGreg.

9. testIndNormLog. This is a second alternative to testIndIGreg.

10. 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.

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

12. testIndQPois. This is a better alternative for discrete target, better than the testIndPois and
    than the testIndNB, because it can capture both cases of overdispersion and undersidpesrion.

13. 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.

14. testIndLogistic. 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.

15. testIndQBinom. This is an alternative to either the testIndLogistic or especially the testInd-
    Beta.

16. testIndMultinom. 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.

17. testIndOrdinal. 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.

18. 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.

19. 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.

20. Alternatives to gSquare. An alternative to the gSquare test is the testIndLogistic. Depend-
    ing on the nature of the target, binary, un-ordered multinomial or ordered multinomial the
    appropriate regression model is fitted.
21. **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.

22. **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.

23. **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.

24. **testIndClogit**. When the data come from a case-control study, the suitable test is via conditional logistic regression.

25. **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.

26. **testIndSPML**. With a circular target, the projected bivariate normal distribution (Presnell et al., 1998) is used to perform regression.

**Tests for clustered/longitudinal data**

1. **testIndGLMMReg, testIndGLMM, testIndGLMMPois & testIndGLMMLogistic**. In the case of a longitudinal or clustered targets (continuous, proportions, binary or counts), the suggested test is via a (generalised) linear mixed model. **testIndGLMMCR** stands for mixed effects Cox regression.

2. **testIndGEEReg, testIndGELogistic, testIndGEEPois, testIndGEENormLog and testIndGEEGamma**. In the case of a longitudinal or clustered targets (continuous, proportions, binary, counts, positive, strictly positive), the suggested test is via GEE (Generalised Estimating Equations).

**Wald based tests**

The available tests for wald.ses and wald.mmpc are listed below. Note, that only continuous predictors are allowed.

<table>
<thead>
<tr>
<th>Target variable</th>
<th>Available tests</th>
<th>Short explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Continuous</td>
<td>waldMMReg</td>
<td>MM regression</td>
</tr>
<tr>
<td>Proportions</td>
<td>waldMMReg</td>
<td>MM regression after logit transformation</td>
</tr>
<tr>
<td>Proportions</td>
<td>waldBeta</td>
<td>Beta regression</td>
</tr>
<tr>
<td>Non negative</td>
<td>waldIGreg</td>
<td>Inverse Gaussian regression</td>
</tr>
<tr>
<td>Strictly positive</td>
<td>waldGamma</td>
<td>Gamma regression</td>
</tr>
<tr>
<td>Non negative</td>
<td>waldNormLog</td>
<td>Gaussian regression with log link</td>
</tr>
<tr>
<td>Successes &amp; totals</td>
<td>testIndBinom</td>
<td>Binomial regression</td>
</tr>
<tr>
<td>Discrete</td>
<td>waldPois</td>
<td>Poisson regression</td>
</tr>
<tr>
<td>Discrete</td>
<td>waldSpeedPois</td>
<td>Poisson regression</td>
</tr>
</tbody>
</table>
### Conditional independence tests

<table>
<thead>
<tr>
<th>Type of Target</th>
<th>Available tests</th>
<th>Short explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discrete</td>
<td>waldZIP</td>
<td>Zero Inflated Poisson regression</td>
</tr>
<tr>
<td>Discrete</td>
<td>waldNB</td>
<td>Negative binomial regression</td>
</tr>
<tr>
<td>Factor with two levels or binary</td>
<td>waldLogistic</td>
<td>Logistic regression</td>
</tr>
<tr>
<td>Factor with more than two levels (ordered)</td>
<td>waldOrdinal</td>
<td>Ordinal logistic regression</td>
</tr>
<tr>
<td>Left censored</td>
<td>waldTobit</td>
<td>Tobit regression</td>
</tr>
<tr>
<td>Case-control</td>
<td>Mixed</td>
<td>testIndClogit Conditional logistic regression</td>
</tr>
<tr>
<td>Survival</td>
<td>waldCR</td>
<td>Cox regression</td>
</tr>
<tr>
<td>Survival</td>
<td>waldWR</td>
<td>Weibull regression</td>
</tr>
<tr>
<td>Survival</td>
<td>waldER</td>
<td>Exponential regression</td>
</tr>
<tr>
<td>Survival</td>
<td>waldLLR</td>
<td>Log-logistic regression</td>
</tr>
</tbody>
</table>

### Permutation based tests

The available tests for perm.ses and perm.mmpc are listed below. Note, that only continuous predictors are allowed.

<table>
<thead>
<tr>
<th>Target variable</th>
<th>Available tests</th>
<th>Short explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Continuous</td>
<td>permFisher</td>
<td>Pearson correlation</td>
</tr>
<tr>
<td>Continuous</td>
<td>permMMFisher</td>
<td>Robust Pearson correlation</td>
</tr>
<tr>
<td>Continuous</td>
<td>permDcor</td>
<td>Distance correlation</td>
</tr>
<tr>
<td>Continuous</td>
<td>permReg</td>
<td>Linear regression</td>
</tr>
<tr>
<td>Proportions</td>
<td>permReg</td>
<td>Linear regression after logit transformation</td>
</tr>
<tr>
<td>Proportions</td>
<td>permBeta</td>
<td>Beta regression</td>
</tr>
<tr>
<td>Non negative</td>
<td>permIGreg</td>
<td>Inverse Gaussian regression</td>
</tr>
<tr>
<td>Strictly positive</td>
<td>permGamma</td>
<td>Gamma regression</td>
</tr>
<tr>
<td>Non negative</td>
<td>permNormLog</td>
<td>Gaussian regression with log link</td>
</tr>
<tr>
<td>Non negative</td>
<td>permWR</td>
<td>Weibull regression</td>
</tr>
<tr>
<td>Successes &amp; totals</td>
<td>permBinom</td>
<td>Binomial regression</td>
</tr>
<tr>
<td>Discrete</td>
<td>permPois</td>
<td>Poisson regression</td>
</tr>
<tr>
<td>Discrete</td>
<td>permZIP</td>
<td>Zero Inflated Poisson regression</td>
</tr>
<tr>
<td>Discrete</td>
<td>permNB</td>
<td>Negative binomial regression</td>
</tr>
<tr>
<td>Factor with two levels or binary</td>
<td>permLogistic</td>
<td>Binary logistic regression</td>
</tr>
<tr>
<td>Factor with more than two levels (nominal)</td>
<td>permMultinom</td>
<td>Multinomial logistic regression</td>
</tr>
<tr>
<td>Factor with more than two levels (ordered)</td>
<td>permOrdinal</td>
<td>Ordinal logistic regression</td>
</tr>
<tr>
<td>Left censored</td>
<td>permTobit</td>
<td>Tobit regression</td>
</tr>
<tr>
<td>Survival</td>
<td>permCR</td>
<td>Cox regression</td>
</tr>
<tr>
<td>Survival</td>
<td>permWR</td>
<td>Weibull regression</td>
</tr>
<tr>
<td>Survival</td>
<td>permER</td>
<td>Exponential regression</td>
</tr>
<tr>
<td>Survival</td>
<td>permLLR</td>
<td>Log-logistic regression</td>
</tr>
</tbody>
</table>
Author(s)

Michail Tsagris <mtsagris@uoc.gr>

References


---

**Conditional independence regression based tests**

*Conditional independence regression based tests*

**Description**

Conditional independence regression based tests.

**Usage**

```r
cond.regs(target, dataset, xIndex, csIndex, test = NULL, wei = NULL, ncores = 1)
```

```r
glmm.condregs(target, reps = NULL, id, dataset, xIndex, csIndex, test, wei = NULL, slopes = FALSE, ncores = 1)
```

```r
ggee.condregs(target, reps = NULL, id, dataset, xIndex, csIndex, test, wei = NULL, correl = "exchangeable", se = "jack", ncores = 1)
```

Arguments

target The target (dependent) variable. It must be a numerical vector.
dataset The independent variable(s). For the "univregs" this can be a matrix or a dataframe with continuous only variables, a data frame with mixed or only categorical variables. For the "wald.univregs", "perm.univregs", "score.univregs" and "rint.regs" this can only by a numerical matrix.
xIndex The indices of the variables whose association with the target you want to test.
csIndex The index or indices of the variable(s) to condition on. If this is 0, the function univregs will be called.
test For the "cond.regs" one of the following: testIndBeta, testIndReg, testIndLogistic, testIndOrdinal, testIndPois, testIndZIP, testIndNB, testIndClogit, testIndBinom, testIndQBinom, testIndIGreg, censIndCR, censIndWR, censIndER, censIndLLR, testIndMMReg, testIndMVreg, testIndMultinom, testIndTobit, testIndGamma, testIndNormLog or testIndSPML for a circular target.
For the "glmm.condregs" one of the following: testIndLMReg, testIndGLMLogistic, testIndGLMMPois, testIndGLMMOrdinal, testIndGLMMGamma or testIndGLMMNormLog.
For the "gee.condregs" one of the following: testIndGEEReg, testIndGEELogistic, testIndGEEPois, testIndGEEGamma or testIndGEENormLog.
Note that in all cases 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. An example where weights are used is surveys when stratified sampling has occured.
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.
id A numerical vector of the same length as target with integer valued numbers, such as 1, 2, 3,... (zeros, negative values and factors are not allowed) specifying the clusters or subjects. This argument is for the rint.regs (see details for more information).
reps If you have measurements over time (logitudinal data) you can put the time here (the length must be equal to the length of the target) or set it equal to NULL. (see details for more information).
slopes Should random slopes for the ime effect be fitted as well? Default value is FALSE.
correl The correlation structure. For the Gaussian, Logistic, Poisson and Gamma regression this can be either "exchangeable" (compound symmetry, suitable for clustered data) or "ar1" (AR(1) model, suitable for longitudinal data).
The method for estimating standard errors. This is very important and crucial. The available options for Gaussian, Logistic, Poisson and Gamma regression are:

- 'san.se', the usual robust estimate.
- 'jack': if approximate jackknife variance estimate should be computed.
- 'j1s': if 1-step jackknife variance estimate should be computed and
- 'fij': logical indicating if fully iterated jackknife variance estimate should be computed.

If you have many clusters (sets of repeated measurements) "san.se" is fine as it is asymptotically correct, plus jackknife estimates will take longer. If you have a few clusters, then maybe it's better to use jackknife estimates.

The jackknife variance estimator was suggested by Paik (1988), which is quite suitable for cases when the number of subjects is small (K < 30), as in many biological studies. The simulation studies conducted by Ziegler et al. (2000) and Yan and Fine (2004) showed that the approximate jackknife estimates are in many cases in good agreement with the fully iterated ones.

Details

This function is more as a help function for 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. The function does not check for zero variance columns, only the "univregs" and related functions do.

If you want to use the GEE methodology, make sure you load the library geepack first.

Value

A list including:

- stat The value of the test statistic.
- pvalue The logarithm of the p-value of the test.

Author(s)

Michail Tsagris

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

References


Conditional independence test for binary, categorical or ordinal data

See Also

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

Examples

```r
y <- rpois(100, 15)
x <- matrix(rnorm(100 * 10), ncol = 10)
a1 <- univregs(y, x, test = testIndPois)
a4 <- cond.regs(y, as.data.frame(x), xIndex = 1:9, csIndex = 10, test = testIndPois)
```

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 p-value 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 appropriate degrees of freedom on the difference between the deviances of the two models.

Usage

```r
testIndLogistic(target, dataset, xIndex, csIndex, wei = NULL, univariateModels = NULL, hash = FALSE, stat_hash = NULL, pvalue_hash = NULL)
testIndMultinom(target, dataset, xIndex, csIndex, wei = NULL, univariateModels = NULL, hash = FALSE, stat_hash = NULL, pvalue_hash = NULL)
testIndOrdinal(target, dataset, xIndex, csIndex, wei = NULL, univariateModels = NULL, hash = FALSE, stat_hash = NULL, pvalue_hash = NULL)
testIndQBinom(target, dataset, xIndex, csIndex, wei = NULL, univariateModels = NULL, hash = FALSE, stat_hash = NULL, pvalue_hash = NULL)
permLogistic(target, dataset, xIndex, csIndex, wei = NULL, univariateModels = NULL, hash = FALSE, stat_hash = NULL, pvalue_hash = NULL, threshold = 0.05, R = 999)
permMultinom(target, dataset, xIndex, csIndex, wei = NULL, univariateModels = NULL, hash = FALSE, stat_hash = NULL, pvalue_hash = NULL, threshold = 0.05, R = 999)
permOrdinal(target, dataset, xIndex, csIndex, wei = NULL, univariateModels = NULL, hash = FALSE, stat_hash = NULL, pvalue_hash = NULL, threshold = 0.05, R = 999)
```

Conditional independence test for binary, categorical or ordinal class variables
threshold = 0.05, R = 999

waldLogistic(target, dataset, xIndex, csIndex, wei = NULL, univariateModels = NULL, hash = FALSE, stat_hash = NULL, pvalue_hash = NULL)

waldQBinom(target, dataset, xIndex, csIndex, wei = NULL, univariateModels = NULL, hash = FALSE, stat_hash = NULL, pvalue_hash = NULL)

waldOrdinal(target, dataset, xIndex, csIndex, wei = NULL, univariateModels = NULL, hash = FALSE, stat_hash = NULL, pvalue_hash = NULL)

Arguments

**target**
A numeric vector containing the values of the target variable. For the "testInd-Logistic" this can either be a binary numerical variable or a factor variable. The factor variable can have two values (binary logistic regression), more than two values (multinomial logistic regression) or it can be an ordered factor with more than two values (ordinal regression). The last one is for example, factor(x, ordered = TRUE). The "waldBinary" is the Wald test version of the binary logistic regression. The "waldOrdinal" is the Wald test version of the ordinal regression.

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. In the cases of "waldBinary", "waldOrdinal" this is strictly a matrix.

xIndex
The index of the variable whose association with the target we want to test.

csIndex
The indices of the variables to condition on. If you have no variables set this equal to 0.

wei
A vector of weights to be used for weighted regression. The default value is NULL. An example where weights are used is surveys when stratified sampling has occured.

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 which contains the cached generated statistics of a SES run in the current dataset, using the current test.

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

threshold
Threshold (suitable values in (0, 1)) for assessing p-values significance.

R
The number of permutations, set to 999 by default. There is a trick to avoid doing all permutations. As soon as the number of times the permuted test statistic is more than the observed test statistic is more than 50 (if threshold = 0.05 and
R = 999), the p-value has exceeded the significance level (threshold value) and hence the predictor variable is not significant. There is no need to continue do the extra permutations, as a decision has already been made.

Details

The testIndQBinom does quasi binomial regression which is suitable for binary targets and proportions including 0 and 1.

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".

The log-likelihood ratio test used in "testIndLogistic" requires the fitting of two models. The Wald test used in waldBinary" and "waldOrdinal" requires fitting of only one model, the full one. The significance of the variable is examined only. Only continuous (or binary) predictor variables are currently accepted in this test.

The testIndQBinom can also be used with percentages, see testIndBeta for example.

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.
- 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 <cathineou@csd.uoc.gr> and Michail Tsagris <mtsagris@uoc.gr>
Conditional independence test for binary, categorical or ordinal data

References


See Also

SES, gSquare, CondIndTests

Examples

#simulate a dataset with categorical data
dataset_m <- matrix( sample(c(0, 1, 2), 20 * 100, replace = TRUE), ncol = 20)
#initialize categorical target
target_m <- dataset_m[, 20]
#remove target from the dataset
dataset_m <- dataset_m[, -20]

#run the conditional independence test for the nominal class variable
testIndMultinom(target_m, dataset_m, xIndex = 14, csIndex = c(1, 2) )

#run the conditional independence test for the ordinal class variable
testIndOrdinal( factor(target_m, ordered = TRUE), dataset_m,
xIndex = 14, csIndex = c(1, 2) )

#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 = "testIndOrdinal")

#simulate a dataset with binary data
dataset_b <- matrix(sample(c(0,1), 50 * 20, replace = TRUE), ncol = 20)
#initialize binary target
target_b <- dataset_b[, 20]
#remove target from the dataset
dataset_b <- dataset_b[, -20]

#run the conditional independence test for the binary class variable
testIndLogistic( target_b, dataset_b, xIndex = 14, csIndex = c(1, 2) )

#run the MMPC algorithm using the testIndLogistic conditional independence test
#for the binary class variable
mmpcObject <- MMPC(target_b, dataset_b, max_k = 3, threshold = 0.05,
test = "testIndLogistic")
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, univariateModels = NULL, hash = FALSE, stat_hash = NULL, pvalue_hash = NULL)

permClogit(target, dataset, xIndex, csIndex, wei = NULL, univariateModels = NULL, hash = FALSE, stat_hash = NULL, pvalue_hash = NULL, threshold = 0.05, R = 999)

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. If you have no variables set this equal to 0.
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 at the conditional logistic regression. See the survival package for more information about conditional logistic regression.
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 successful) 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 which contains the cached generated statistics of a SES run in the current dataset, using the current test.

pvalue_hash

A hash object which contains the cached generated p-values of a SES run in the current dataset, using the current test.

threshold

Threshold (suitable values in (0, 1)) for assessing p-values significance.

R

The number of permutations, set to 999 by default. There is a trick to avoid doing all permutations. As soon as the number of times the permuted test statistic is more than the observed test statistic is more than 50 (if threshold = 0.05 and R = 999), the p-value has exceeded the significance level (threshold value) and hence the predictor variable is not significant. There is no need to continue do the extra permutations, as a decision has already been made.

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 log-likelihood for a conditional logistic regression model equals the log-likelihood 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".

Note that this function is a bit sensitive and prone to breaking down for some reason which I have not yet figured out. In case of singular design matrix it will work, it identifies the problem and handles it internally like most regression functions do. However, problems can still occur.

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).
Conditional independence test for circular 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. The p-value 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 log-likelihood ratio test using circular regression with 2 degrees of freedom.

- **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@uoc.gr>

**References**


**See Also**

*SES, testIndLogistic, censIndCR, censIndWR*

**Examples**

```r
# simulate a dataset with continuous data
data <- matrix(rnorm(100 * 7), nrow = 100)
# the target feature is the last column of the dataset as a vector
case <- rbinom(100, 1, 0.6)
ing <- which(case == 1)
ing <- sample(in, 50)
case[-in] = 0
id <- rep(1:50, 2)
target <- cbind(case, id)

results <- testIndClogit(target, dataset, xIndex = 4, csIndex = 1)
results

# run the SES algorithm using the testIndClogit conditional independence test
a <- MMPC(target, dataset, max_k = 3, threshold = 0.01, test = "testIndClogit")
```

---

Circular regression conditional independence test for circular class dependent variables and continuous predictors.
Usage

testIndSPML(target, dataset, xIndex, csIndex, wei = NULL, univariateModels = NULL, hash = FALSE, stat_hash = NULL, pvalue_hash = NULL)

Arguments

target A numerical vector with the data expressed in radians, or a 2 column matrix with the cosine and sine of the response (i.e. unit vector). See examples for more information on this.
dataset A numeric matrix with the variables for performing the test. Rows as samples and columns as features. Currently this test accepts only continuous variables.
xIndex The index of the variable whose association with the target we want to test.
csIndex The indices of the variables to condition on. If you have no variables set this equal to 0.
wei This is not used in this test.
univariateModels Fast alternative to the hash object for univariate test. List with vectors "pvalues" (p-values), "stats" (statistics) 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 statistic test. Default value is FALSE. If TRUE you have to specify the stat_hash argument and the pvalue_hash argument.
stat_hash A hash object which contains the cached generated statistics of a SES run in the current dataset, using the current test.
pvalue_hash A hash object which contains the cached generated p-values of a SES run in the current dataset, using the current test.

Details

If hash = TRUE, testIndSPML 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".

The log-likelihood ratio test used in "testIndSPML" requires the fitting of two models. The significance of the variable is examined and only continuous (or binary) predictor variables are currently accepted in this test.

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).
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 logged p-values. See argument stat_hash and details. If argument hash = FALSE this is NULL.

Author(s)
Michail Tsagris
R implementation and documentation: Michail Tsagris <mtsagris@uoc.gr>

References

See Also
univregs, SES, testIndReg, CondIndTests

Examples
y <- runif(100, -pi, pi)  ## suppose these are radians
x <- matrix( rnorm(100 * 3), ncol = 3)
testIndSPML(y1, x, csIndex = 1, xIndex = 2)
## alternatively
y1 <- cbind(cos(y), sin(y))  ## a matrix with two columns
testIndSPML(y1, x, csIndex = 1, xIndex = 2)

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

testIndGEEReg(target, reps = NULL, group, dataset, xIndex, csIndex, wei = NULL, univariateModels = NULL, hash = FALSE, stat_hash = NULL, pvalue_hash = NULL, correl = "exchangeable", se = "jack")

testIndGEELogistic(target, reps = NULL, group, dataset, xIndex, csIndex, wei = NULL, univariateModels = NULL, hash = FALSE, stat_hash = NULL, pvalue_hash = NULL, correl = "exchangeable", se = "jack")

testIndGEEPois(target, reps = NULL, group, dataset, xIndex, csIndex, wei = NULL, univariateModels = NULL, hash = FALSE, stat_hash = NULL, pvalue_hash = NULL, correl = "exchangeable", se = "jack")

testIndGEEGamma(target, reps = NULL, group, dataset, xIndex, csIndex, wei = NULL, univariateModels = NULL, hash = FALSE, stat_hash = NULL, pvalue_hash = NULL, correl = "exchangeable", se = "jack")

testIndGEENormLog(target, reps = NULL, group, dataset, xIndex, csIndex, wei = NULL, univariateModels = NULL, hash = FALSE, stat_hash = NULL, pvalue_hash = NULL, correl = "exchangeable", se = "jack")

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 length as target.

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. If you have no variables set this equal to 0.

wei A vector of weights to be used for weighted regression. The default value is NULL. It is mentioned in the “geepack” that weights is not (yet) the weight as in sas proc genmod, and hence is not recommended to use.

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 which contains the cached generated statistics of a SES run in the current dataset, using the current test.

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

correl
The correlation structure. For the Gaussian, Logistic, Poisson and Gamma regression this can be either "exchangeable" (compound symmetry, suitable for clustered data) or "ar1" (AR(1) model, suitable for longitudinal data). For the ordinal logistic regression its only the "exchangeable" correlation structure.

se
The method for estimating standard errors. This is very important and crucial. The available options for Gaussian, Logistic, Poisson and Gamma regression are: a) 'san.se', the usual robust estimate. b) 'jack': if approximate jackknife variance estimate should be computed. c) 'j1s': if 1-step jackknife variance estimate should be computed and d) 'fij': logical indicating if fully iterated jackknife variance estimate should be computed. If you have many clusters (sets of repeated measurements) "san.se" is fine as it is asymptotically correct, plus jackknife estimates will take longer. If you have a few clusters, then maybe it’s better to use jackknife estimates.

The jackknife variance estimator was suggested by Paik (1988), which is quite suitable for cases when the number of subjects is small (K < 30), as in many biological studies. The simulation studies conducted by Ziegler et al. (2000) and Yan and Fine (2004) showed that the approximate jackknife estimates are in many cases in good agreement with the fully iterated ones.

Details
If hash = TRUE, testIndGEE 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 higher order as well, but this would be a hyper-parameter, increasing the complexity of the models to be tested.

Make sure you load the library geepack first.

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).

**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)**

Michail Tsagris

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

**References**


**See Also**

`SES.glmm`, `MMPC.glmm`, `CondIndTests`

**Examples**

```r
library("geepack", quietly = TRUE)
y <- rnorm(150)
x <- matrix(rnorm(150 * 5), ncol = 5)
id <- sample(1:20, 150, replace = TRUE)
testIndGEEReg(y, group = id, dataset = x, xIndex = 1, csIndex = 3)
```
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

testIndGLMMReg(target, reps = NULL, group, dataset, xIndex, csIndex, wei = NULL, univariateModels = NULL, hash = FALSE, stat_hash = NULL, pvalue_hash = NULL, slopes = FALSE)

testIndGLMMLogistic(target, reps = NULL, group, dataset, xIndex, csIndex, wei = NULL, univariateModels = NULL, hash = FALSE, stat_hash = NULL, pvalue_hash = NULL, slopes = FALSE)

testIndGLMMPois(target, reps = NULL, group, dataset, xIndex, csIndex, wei = NULL, univariateModels = NULL, hash = FALSE, stat_hash = NULL, pvalue_hash = NULL, slopes = FALSE)

testIndGLMMNB(target, reps = NULL, group, dataset, xIndex, csIndex, wei = NULL, univariateModels = NULL, hash = FALSE, stat_hash = NULL, pvalue_hash = NULL, slopes = FALSE)

testIndGLMMGamma(target, reps = NULL, group, dataset, xIndex, csIndex, wei = NULL, univariateModels = NULL, hash = FALSE, stat_hash = NULL, pvalue_hash = NULL, slopes = FALSE)

testIndGLMMNormLog(target, reps = NULL, group, dataset, xIndex, csIndex, wei = NULL, univariateModels = NULL, hash = FALSE, stat_hash = NULL, pvalue_hash = NULL, slopes = FALSE)

testIndGLMMOrdinal(target, reps = NULL, group, dataset, xIndex, csIndex, wei = NULL, univariateModels = NULL, hash = FALSE, stat_hash = NULL, pvalue_hash = NULL, slopes = FALSE)

testIndGLMCMR(target, reps = NULL, group, dataset, xIndex, csIndex, wei = NULL, univariateModels = NULL, hash = FALSE, stat_hash = NULL, pvalue_hash = NULL, slopes = FALSE)
testIndLMM(target, reps = NULL, group, dataset, xIndex, csIndex, wei = NULL, univariateModels = NULL, hash = FALSE, stat_hash = NULL, pvalue_hash = NULL, 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. In the case of "testIndGLMMOrdinal" this must be an ordered factor.

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. This is not applied in ordinal and Cox regression.

group A numeric vector containing the subjects or groups. It must be of the same length as target.

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. If you have no variables set this equal to 0.

wei A vector of weights to be used for weighted regression. The default value is NULL. An example where weights are used is surveys when stratified sampling has occured.

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 which contains the cached generated statistics of a SES run in the current dataset, using the current test.

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

slopes A boolean variable which indicates whether (TRUE) to or not (FALSE) random slopes in the time effect as well. By default random intercepts are considered. This is not applied in ordinal and Cox regression.
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 higher order as well, but this would be a hyper-parameter, increasing the complexity of the models to be tested.

The testIndLMM is used for linear mixed models with no weights, no slopes and no reps. This is a random intercepts model only. The advantage of this function is that it is tens of times faster than testIndGLMM.

Value

A list including:

- \texttt{pvalue} : A numeric value that represents the logarithm of the generated p-value due to the (generalised) linear mixed model (see reference below).
- \texttt{stat} : A numeric value that represents the generated statistic due to the (generalised) linear mixed model (see reference below).
- \texttt{stat_hash} : The current hash object used for the statistics. See argument stat_hash and details. If argument hash = FALSE this is NULL.
- \texttt{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@uoc.gr>

References


See Also

SES.glmm, MMPC.glmm, CondIndTests
Examples

```r
y <- rnorm(150)
x <- matrix(rnorm(150 * 5), ncol = 5)
id <- sample(1:20, 150, replace = TRUE)
testIndGLMMReg(y, group = id, dataset = x, xIndex = 1, csIndex = 3)
testIndLMM(y, group = id, dataset = x, xIndex = 1, csIndex = 3)
```

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

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

```r
permBeta(target, dataset, xIndex, csIndex, wei = NULL, univariateModels = NULL, hash = FALSE, stat_hash = NULL, pvalue_hash = NULL, threshold = 0.05, R = 999)
```

```r
waldBeta(target, dataset, xIndex, csIndex, wei = NULL, univariateModels = NULL, hash = FALSE, stat_hash = NULL, pvalue_hash = NULL)
```

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. In the case of "waldBeta" this is strictly a matrix.

**xIndex**
The index of the variable whose association with the target we want to test.

**csIndex**
The indices of the variables to condition on. If you have no variables set this equal to 0.

**wei**
A vector of weights to be used for weighted regression. The default value is NULL. An example where weights are used is surveys when stratified sampling has occured.
Conditional independence test for proportions/percentages

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 successful) 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 which contains the cached generated statistics of a SES run in the current dataset, using the current test.

pvalue_hash

A hash object which contains the cached generated p-values of a SES run in the current dataset, using the current test.

threshold

Threshold (suitable values in (0, 1)) for assessing p-values significance.

R

The number of permutations, set to 999 by default. There is a trick to avoid doing all permutations. As soon as the number of times the permuted test statistic is more than the observed test statistic is more than 50 (if threshold = 0.05 and R = 999), the p-value has exceeded the significance level (threshold value) and hence the predictor variable is not significant. There is no need to continue doing the extra permutations, as a decision has already been made.

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.

The log-likelihood ratio test used in "testIndBeta" requires the fitting of two models. The Wald test used in "waldBeta" requires fitting of only one model, the full one. The significance of the variable is examined only. Only continuous (or binary) predictor variables are currently accepted in this test.

The testIndQBinom is another alternative to use.

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).

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 logged 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@uoc.gr>

References


See Also

testIndQBinom, SES, testIndReg, testIndRQ, testIndFisher, CondIndTests

Examples

#simulate a dataset with continuous data
dataset <- matrix(runif(100 * 10, 1, 1000), ncol = 10)
#the target feature is the last column of the dataset as a vector
target <- dataset[, 10]
dataset <- dataset[, -10]
target <- target / (max(target) + 2)
testIndBeta(target, dataset, xIndex = 4, csIndex = 9)

#run the MMPC algorithm using the testIndBeta conditional independence test
mmpcObject <- MMPC(target, dataset, max_k = 3, threshold = 0.05, test = "testIndBeta")

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 apropriate degrees of freedom on the difference between the deviances of the two models.
Usage

\[
\text{testIndTimeLogistic}(\text{target}, \text{dataset}, \text{xIndex}, \text{csIndex}, \text{wei = NULL,}}
\text{univariateModels = NULL, hash = FALSE, stat_hash = NULL, pvalue_hash = NULL})
\]

\[
\text{testIndTimeMultinom}(\text{target}, \text{dataset}, \text{xIndex}, \text{csIndex}, \text{wei = NULL,}}
\text{univariateModels = NULL, hash = FALSE, stat_hash = NULL, pvalue_hash = NULL})
\]

Arguments

target A numeric vector containing the values of the target variable. For the "testIndLogistic" this can either be a binary numerical variable or a factor variable. The factor variable can have two values (binary logistic regression), more than two values (multinomial logistic regression) or it can be an ordered factor with more than two values (ordinal regression). The last one is for example, factor(x, ordered = TRUE). The "waldBinary" is the Wald test version of the binary logistic regression. The "waldOrdinal" is the Wald test version of the ordinal regression.
dataset A numeric matrix with the constants and slopes stack one upo the other. The first r rows are the constants and the rest of the rows contains the slopes. In some the matrix can be calculated using the group.mvbetas function.
xIndex The index of the variable whose association with the target we want to test.
csIndex The indices of the variables to condition on. If you have no variables set this equal to 0.
wei A vector of weights to be used for weighted regression. The default value is NULL. An example where weights are used is surveys when stratified sampling has occurred.
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 which contains the cached generated statistics of a SES run in the current dataset, using the current test.
pvalue_hash A hash object which contains the cached generated p-values of a SES run in the current dataset, using the current test.

Details

This conditional independence test is devised for the static-longitudinal scenario of Tsagris, Lagani and Tsamardinos (2018). The idea is that you have many features of longitudinal data for many subjects. For each subject you have calculated the coefficients of a simple linear regression over time and this is repeated for each feature. In the end, assuming p features, you have p constants and p slopes for each subject, each constant and slope refers to a feature for a subject.
Conditional independence test for the static-longitudinal scenario

**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.
- **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)**

Michail Tsagris

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

**References**


**See Also**

`SES`, `gSquare`, `CondIndTests`

**Examples**

```r
## assume these are longitudinal data, each column is a variable (or feature)
x <- matrix(rnorm(400 * 50), ncol = 50)
id <- rep(1:80, each = 5)  ## 80 subjects
reps <- rep(seq(4, 12, by = 2), 80)  ## 5 time points for each subject
dataset <- group.mvbetas(x, id, reps)
## these are the regression coefficients of the first subject's values on the
## reps (which is assumed to be time in this example)
target <- rbinom(80, 1, 0.5)
testIndTimeLogistic(target, dataset, xIndex = 1, csIndex = 0)
testIndTimeLogistic(target, dataset, xIndex = 1, csIndex = 2)
```
Description

Many conditional independence tests counting the number of times a possible collider d-separates two nodes.

Usage

condis(ind1, ind2, cs1, cs2, Var, dat, type = "pearson", rob = FALSE, max_k = 2, R = 1)

Arguments

ind1
The index of the one variable to be considered.

ind2
The index of the other variable to be considered.

cs1
The index or indices of the conditioning set of variable(s). These are the neighbours of node ind1.

cs2
The index or indices of the conditioning set of variable(s). These are the neighbours of node ind2.

Var
The index of the possible collider.

dat
A numerical matrix or a data.frame with numerical, binary, nominal and ordinal variables only.

type
This is either "pearson", "spearman", "cat", "distcor", "ci.mm" or "ci.fast".

rob
In case you want robust estimation of the Pearson correlation prior to applying the conditional independence test. This is activated only when type = "pearson".

max_k
The maximum number of conditioning variables to consider. It can be the case that each node ind1 and ind2 has 10 neighbours. We should try all possible combinations of the neighbours of ind1 and then of ind2. To reduce the computational cost we search the subsets with at most max_k variables.

R
This is used by most tests, except for type = "ci.mm" and type = "ci.fast".

Details

This is to be used in the conservative version of Rule 0 of the Pc algorithm. When one wants to know whether a variable is a possible collider, Ramsey, Spirtes and Zhang (2005) propose to perform the following action. For every unshielded triple (X, Y, Z) check all subsets of X’s possible parents and of Z’s possible parents. a) If Y is NOT in any such set conditional on which, X and Z are independent orient X - Y - Z as X -> Y <- Z. b) If Y is in ALL sets conditional on which, X and Z are independent, leave X - Y - Z as it is, i.e. a non-collider. c) Mark the triple X - Y - Z as "unfaitfull" otherwise. This modification leads to the so called conservative PC (CPC) algorithm.
Conditional independence tests counting the number of times a possible collider d-separates two nodes

A few years later, Colombo and Maathuis (2014) suggested a modification of the previous action, called the majority rule. If \( Y \) is less than 50\% of the sets that render \( X \) and \( Z \) independent orient \( X - Y - Z \) as \( X \to Y \leftarrow Z \). If \( Y \) is found in exactly 50\% of sets that render \( X \) and \( Z \) independent, this triple is marked as "ambiguous". This modification leads the so called majority rule PC (MPC) algorithm.

This function we have implemented here, does exactly this. It applies tests to many subsets and returns a matrix with two columns. The first one contains 0 or 1 and the second is the p-value. A value of 0 indicates absence of the possible collider from the set that produced that p-value, whereas a value of 1 indicates its presence in the set.

This way, we can measure the proportion of times the possible collider \( Y \) was in a subset that rendered \( X \) and \( Z \) independent.

Value

A matrix with two columns. The second one is the logarithm of the p-value. The first one contains 0s and 1s. The value of 0 means that the candidate collider was not in that set which produced the relevant p-value, whereas a value of 1 indicates that it was a member of that conditioning set.

Author(s)

Michail Tsagris

R implementation and documentation: Michail Tsagris &lt;mtsagris@uoc.gr&gt;

References


See Also

SES, MMPC, testIndLogistic

Examples

```R
x <- rdag2(1000, p = 10, nei = 5)
G <- x$G
dat <- x$x
cs1 <- which(G[6, ] > 0 | G[, 6] > 0)
cs2 <- which(G[7, ] > 0 | G[, 7] > 0)
cs1 <- setdiff( cs1, c(7, 3) )
cs2 <- setdiff( cs2, c(6, 3) )
condis(6, 7, cs1, cs2, 3, dat, type = "pearson", rob = FALSE, max_k = 3, R = 1 )
```
Conditional independence tests for continuous univariate and multivariate 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. 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, univariateModels = NULL, hash = FALSE, stat_hash = NULL, pvalue_hash = NULL)

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

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

testIndMMReg(target, dataset, xIndex, csIndex, wei = NULL, univariateModels = NULL, hash = FALSE, stat_hash = NULL, pvalue_hash = NULL)

waldMMReg(target, dataset, xIndex, csIndex, wei = NULL, univariateModels = NULL, hash = FALSE, stat_hash = NULL, pvalue_hash = NULL)

permReg(target, dataset, xIndex, csIndex, wei = NULL, univariateModels = NULL, hash = FALSE, stat_hash = NULL, pvalue_hash = NULL, threshold = 0.05, R = 999)

permMMReg(target, dataset, xIndex, csIndex, wei = NULL, univariateModels = NULL, hash = FALSE, stat_hash = NULL, pvalue_hash = NULL, threshold = 0.05, R = 999)

permRQ(target, dataset, xIndex, csIndex, wei = NULL, univariateModels = NULL, hash = FALSE, stat_hash = NULL, pvalue_hash = NULL, threshold = 0.05, R = 999)

permMVreg(target, dataset, xIndex, csIndex, wei = NULL, univariateModels = NULL, hash = FALSE, stat_hash = NULL, pvalue_hash = NULL, threshold = 0.05, 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 \( \log(\frac{target}{1 - target}) \). 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. In the cases of "waldIGreg" and "waldMMreg" this is strictly a matrix.

**xIndex**
The index of the variable whose association with the target we want to test.

**csIndex**
The indices of the variables to condition on. If you have no variables set this equal to 0.

**wei**
A vector of weights to be used for weighted regression. The default value is NULL. An example where weights are used is surveys when stratified sampling has occurred. They are not take into account in the robust regression via MM estimation (testIndMMReg and permMMreg).

**univariateModels**
Fast alternative to the hash object for univariate test. List with vectors "pvalues" (p-values) and "stats" (statistics) 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 which contains the cached generated statistics of a SES run in the current dataset, using the current test.

**pvalue_hash**
A hash object which contains the cached generated p-values of a SES run in the current dataset, using the current test.

**threshold**
Threshold (suitable values in (0, 1)) for assessing p-values significance.

**R**
The number of permutations, set to 999 by default. There is a trick to avoid doing all permutations. As soon as the number of times the permuted test statistic is more than the observed test statistic is more than 50 (if threshold = 0.05 and \( R = 999 \)), the p-value has exceeded the significance level (threshold value) and hence the predictor variable is not significant. There is no need to continue do the extra permutations, as a decision has already been made.

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 regression.
testIndMMReg offers robust linear MM 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.  
For all the available conditional independence tests that are currently included on the package, please see "\texttt{?CondIndTests}".  
The Wald test used in "\texttt{waldMMReg}" requires fitting of only one model, the full one. The significance of the variable is examined only. Only continuous (or binary) predictor variables are currently accepted in this test.  

\textbf{Value}  
A list including:  

\begin{itemize}  
\item \texttt{pvalue} A numeric value that represents the logarithm of the generated p-value due to linear regression (see reference below).  
\item \texttt{stat} A numeric value that represents the generated statistic due to linear regression (see reference below).  
\item \texttt{stat\_hash} The current hash object used for the statistics. See argument \texttt{stat\_hash} and details. If argument hash = FALSE this is NULL.  
\item \texttt{pvalue\_hash} The current hash object used for the p-values. See argument \texttt{stat\_hash} and details. If argument hash = FALSE this is NULL.  
\end{itemize}  

\textbf{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@uoc.gr>  

\textbf{References}  
Conditional independence tests for count data

See Also

testIndRQ, testIndFisher, testIndSpearman, CondIndTests

Examples

# simulate a dataset with continuous data
dataset <- matrix(runif(100 * 50, 1, 100), ncol = 50)
# the target feature is the last column of the dataset as a vector
target <- dataset[, 50]
dataset <- dataset[, -50]

testIndReg(target, dataset, xIndex = 44, csIndex = 10)
testIndMMReg(target, dataset, xIndex = 44, csIndex = 10)
testIndRQ(target, dataset, xIndex = 44, csIndex = 10)
testIndIGreg(target, dataset, xIndex = 44, csIndex = 10)

# run the MMPC algorithm using the testIndReg conditional independence test
m1 <- MMPC(target, dataset, max_k = 3, threshold = 0.05, test = "testIndReg")
m2 <- MMPC(target, dataset, max_k = 3, threshold = 0.05, test = "testIndRQ")

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 PV ALUE 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 inflated poisson and negative binomial.

Usage

testIndPois(target, dataset, xIndex, csIndex, wei = NULL, univariateModels = NULL, hash = FALSE, stat_hash = NULL, pvalue_hash = NULL)
testIndNB(target, dataset, xIndex, csIndex, wei = NULL, univariateModels = NULL, hash = FALSE, stat_hash = NULL, pvalue_hash = NULL)
testIndZIP(target, dataset, xIndex, csIndex, wei = NULL, univariateModels = NULL, hash = FALSE, stat_hash = NULL, pvalue_hash = NULL)
testIndQPois(target, dataset, xIndex, csIndex, wei = NULL, univariateModels = NULL, hash = FALSE, stat_hash = NULL, pvalue_hash = NULL)
Conditional independence tests for count data

permPois(target, dataset, xIndex, csIndex, wei = NULL, univariateModels = NULL, hash = FALSE, stat_hash = NULL, pvalue_hash = NULL, threshold = 0.05, R = 999)

permNB(target, dataset, xIndex, csIndex, wei = NULL, univariateModels = NULL, hash = FALSE, stat_hash = NULL, pvalue_hash = NULL, threshold = 0.05, R = 999)

permZIP(target, dataset, xIndex, csIndex, wei = NULL, univariateModels = NULL, hash = FALSE, stat_hash = NULL, pvalue_hash = NULL, threshold = 0.05, R = 999)

waldPois(target, dataset, xIndex, csIndex, wei = NULL, univariateModels = NULL, hash = FALSE, stat_hash = NULL, pvalue_hash = NULL)

waldNB(target, dataset, xIndex, csIndex, wei = NULL, univariateModels = NULL, hash = FALSE, stat_hash = NULL, pvalue_hash = NULL)

waldZIP(target, dataset, xIndex, csIndex, wei = NULL, univariateModels = NULL, hash = FALSE, stat_hash = NULL, pvalue_hash = NULL)

Arguments

target: A numeric vector containing the values of the target variable.
dataset: A numeric matrix or data frame, in case of categorical predictors (factors), containing the variables for performing the test. Rows as samples and columns as features. In the cases of "waldPois", "waldNB" and "waldZIP" this is strictly a matrix.
xIndex: The index of the variable whose association with the target we want to test.
csIndex: The indices of the variables to condition on. If you have no variables set this equal to 0.
wei: A vector of weights to be used for weighted regression. The default value is NULL. An example where weights are used is surveys when stratified sampling has occurred.
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 which contains the cached generated statistics of a SES run in the current dataset, using the current test.
pvalue_hash: A hash object which contains the cached generated p-values of a SES run in the current dataset, using the current test.
threshold  Threshold (suitable values in (0, 1)) for assessing p-values significance.
R  The number of permutations, set to 999 by default. There is a trick to avoid doing all permutations. As soon as the number of times the permuted test statistic is more than the observed test statistic is more than 50 (if threshold = 0.05 and R = 999), the p-value has exceeded the significance level (threshold value) and hence the predictor variable is not significant. There is no need to continue do the extra permutations, as a decision has already been made.

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 "SESoutpu"t' 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 inflated 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.

The log-likelihood ratio test used in "testIndPois", "testIndNB" and "testIndZIP" requires the fitting of two models. The Wald test used in "waldPois", "waldNB" and "waldZIP" requires fitting of only one model, the full one. The significance of the variable is examined only. Only continuous (or binary) predictor variables are currently accepted in this test.

The testIndQPois does quasi Poisson regression. The benefit of this regression is that it works for over and under dispersed data. Negative Binomial works for over dispersed data, but not for under dispersed data. In addition it is fast.

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).
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@uoc.gr>
### References


### See Also

`testIndReg, testIndNB, testIndZIP, gSquare, CondIndTests`

### Examples

```r
# simulate a dataset with continuous data
dataset <- matrix(runif(100 * 20, 1, 50), ncol = 20)
# the target feature is the last column of the dataset as a vector
target <- rpois(100, 10)
results <- testIndPois(target, dataset, xIndex = 14, csIndex = 10)
results
```

```r
# run the SES algorithm using the testIndPois conditional independence test
m1 <- MMPC(target, dataset, max_k = 3, threshold = 0.05, test = "testIndPois");
m2 <- MMPC(target, dataset, max_k = 3, threshold = 0.05, test = "testIndNB");
```

### Conditional independence tests for left censored 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 be based on the Cox (semiparametric) regression or on the Weibull (parametric) regression.

### Usage

```r
testIndTobit(target, dataset, xIndex, csIndex, wei = NULL,
univariateModels = NULL, hash = FALSE, stat_hash = NULL, pvalue_hash = NULL)
```

```r
waldTobit(target, dataset, xIndex, csIndex, wei = NULL,
univariateModels = NULL, hash = FALSE, stat_hash = NULL, pvalue_hash = NULL)
```

```r
permTobit(target, dataset, xIndex, csIndex, wei = NULL,
univariateModels = NULL, hash = FALSE, stat_hash = NULL, pvalue_hash = NULL,
threshold = 0.05, R = 999)
```
Arguments

**target**
A Survival object (class Surv from package survival) with left censored data. This test works with responses that are left censored. See the examples below (the command `Surv` or the final example in the `survreg` documentation of the "survival" package) for more information on how to create 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. In the cases of "waldTobit" and "permTobit" this is strictly a matrix.

**xIndex**
The index of the variable whose association with the target we want to test.

**csIndex**
The indices of the variables to condition on. If you have no variables set this equal to 0.

**wei**
A vector of weights to be used for weighted regression. The default value is NULL. An example where weights are used is surveys when stratified sampling has occurred.

**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 successful) 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 which contains the cached generated statistics of a SES run in the current dataset, using the current test.

**pvalue_hash**
A hash object which contains the cached generated p-values of a SES run in the current dataset, using the current test.

**threshold**
Threshold (suitable values in (0, 1)) for assessing p-values significance.

**R**
The number of permutations, set to 999 by default. There is a trick to avoid doing all permutations. As soon as the number of times the permuted test statistic is more than the observed test statistic is more than 50 (in this example case), the p-value has exceeded the significance level (threshold value) and hence the predictor variable is not significant. There is no need to continue do the extra permutations, as a decision has already been made.

Details

Tobit regression is performed. The implied model is Gaussian with left censored data.

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".

The log-likelihood ratio test used in "testIndTobit" requires the fitting of two models. The Wald test used in "waldTobit", requires fitting of only one model, the full one. The significance of the variable is examined only. Only continuous (or binary) predictor variables are currently accepted in this test.

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.
- **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 "survreg" and Surv of the package survival and the function anova (analysis of variance) of the package stats.

Author(s)

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

References


See Also

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

Examples

```r
require(survival, quietly = TRUE)

x <- matrix( rnorm(100 * 30), ncol = 30)
y <- x[, 1] - x[, 2] + rnorm(100, 5)
y[y < 0 ] <- 0
y <- survival::Surv(y, y>0, type = 'left')

#run the censIndCR conditional independence test
testIndTobit(y, x, xIndex = 12, csIndex = c(5, 7, 4) )
waldTobit(y, x, xIndex = 12, csIndex = c(5, 7, 4) )
permTobit(y, x, xIndex = 12, csIndex = c(5, 7, 4), R = 499 )
```
#run the SES algorithm using the censIndCR conditional independence
#test for the survival class variable
a <- MMPC(y, x, max_k = 2, threshold = 0.05, test = "testIndTobit")

Conditional independence tests for positive data

Regression conditional independence test for positive 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 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 inflated poisson and negative binomial.

Usage

testIndGamma(target, dataset, xIndex, csIndex, wei = NULL,
univariateModels = NULL, hash = FALSE, stat_hash = NULL, pvalue_hash = NULL)

testIndNormLog(target, dataset, xIndex, csIndex, wei = NULL,
univariateModels = NULL, hash = FALSE, stat_hash = NULL, pvalue_hash = NULL)

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

permGamma(target, dataset, xIndex, csIndex, wei = NULL,
univariateModels = NULL, hash = FALSE, stat_hash = NULL, pvalue_hash = NULL,
threshold = 0.05, R = 999)

permNormLog(target, dataset, xIndex, csIndex, wei = NULL,
univariateModels = NULL, hash = FALSE, stat_hash = NULL, pvalue_hash = NULL,
threshold = 0.05, R = 999)

permIGreg(target, dataset, xIndex, csIndex, wei = NULL,
univariateModels = NULL, hash = FALSE, stat_hash = NULL, pvalue_hash = NULL,
threshold = 0.05, R = 999)

waldGamma(target, dataset, xIndex, csIndex, wei = NULL,
univariateModels = NULL, hash = FALSE, stat_hash = NULL, pvalue_hash = NULL)

waldNormLog(target, dataset, xIndex, csIndex, wei = NULL,
univariateModels = NULL, hash = FALSE, stat_hash = NULL, pvalue_hash = NULL)
waldIGreg(target, dataset, xIndex, csIndex, wei = NULL, univariateModels = NULL, hash = FALSE, stat_hash = NULL, pvalue_hash = NULL)

Arguments

target A numeric vector containing the values of the target variable. For the Gamma based tests, the values must be strictly greater than zero. For the NormLog case, zeros can be included.
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. In the cases of "waldPois", "waldNB" and "waldZIP" this is strictly a matrix.
xIndex The index of the variable whose association with the target we want to test.
csIndex The indices of the variables to condition on. If you have no variables set this equal to 0.
wei A vector of weights to be used for weighted regression. The default value is NULL. An example where weights are used is surveys when stratified sampling has occured.
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 which contains the cached generated statistics of a SES run in the current dataset, using the current test.
pvalue_hash A hash object which contains the cached generated p-values of a SES run in the current dataset, using the current test.
threshold Threshold (suitable values in (0, 1)) for assessing p-values significance.
R The number of permutations, set to 999 by default. There is a trick to avoid doing all permutations. As soon as the number of times the permuted test statistic is more than the observed test statistic is more than 50 (if threshold = 0.05 and R = 999), the p-value has exceeded the significance level (threshold value) and hence the predictor variable is not significant. There is no need to continue do the extra permutations, as a decision has already been made.

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.
Conditional independence tests for positive data

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".

For the testIndGamma and testIndNormLog the F test is used and not the log-likelihood ratio test because both of these regression models have a nuisance parameter. The testIndNormLog can be seen as a non linear Gaussian model where the conditional mean is related with the covariate(s) via an exponential function.

TestIndIGreg fits an inverse gaussian distribution with a log link. The testIndIGreg has some problems due to problems in R’s implementation of the inverse gaussian regression with a log link.

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).
- **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)

Michail Tsagris

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

References


See Also

testIndReg, testIndNB, testIndZIP, gSquare, CondIndTests

Examples

```r
#simulate a dataset with continuous data
dataset <- matrix( rnorm(200 * 20, 1, 5), ncol = 20 )
#the target feature is the last column of the dataset as a vector
target <- rgamma(200, 1, 3)
testIndGamma(target, dataset, xIndex = 14, csIndex = 10)
testIndNormLog(target, dataset, xIndex = 14, csIndex = 10)
#run the MMPC algorithm using the testIndPois conditional independence test
m1 <- MMPC(target, dataset, max_k = 3, threshold = 0.05, test = "testIndGamma");
m2 <- MMPC(target, dataset, max_k = 3, threshold = 0.05, test = "testIndNormLog");
```
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 p-value 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, univariateModels = NULL, hash = FALSE, stat_hash = NULL, pvalue_hash = NULL)

permBinom(target, dataset, xIndex, csIndex, wei = NULL, univariateModels = NULL, hash = FALSE, stat_hash = NULL, pvalue_hash = NULL, threshold = 0.05, R = 999)

waldBinom(target, dataset, xIndex, csIndex, wei = NULL, univariateModels = NULL, hash = FALSE, stat_hash = NULL, pvalue_hash = NULL)

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. In the case of "waldBinom" this is strictly a matrix.
- **xIndex**: The index of the variable whose association with the target we want to test.
- **csIndex**: The indices of the variables to condition on. If you have no variables set this equal to 0.
- **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.
- **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 successful) 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 which contains the cached generated statistics of a SES run in the current dataset, using the current test.

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

threshold Threshold (suitable values in (0, 1)) for assessing p-values significance.

R The number of permutations, set to 999 by default. There is a trick to avoid doing all permutations. As soon as the number of times the permuted test statistic is more than the observed test statistic is more than 50 (if threshold = 0.05 and R = 999), the p-value has exceeded the significance level (threshold value) and hence the predictor variable is not significant. There is no need to continue doing the extra permutations, as a decision has already been made.

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 "SEOutput" is the output of a SES run, then these objects can be retrieved by SEOutput@hashObject$stat_hash and the SEOutput@hashObject$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 inflated Poisson is to be used. This is in fact a logistic regression where the target is the ratio of successes divided by the totals and the weights are the totals.

The log-likelihood ratio test used in "testIndBinom" requires the fitting of two models. The Wald test used in "waldBinom" requires fitting of only one model, the full one. The significance of the variable is examined only. Only continuous (or binary) predictor variables are currently accepted in this test.

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).

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@uoc.gr>

References


See Also

testIndLogistic, testIndBeta, testIndReg, CondIndTests

Examples

# simulate a dataset with continuous data
dataset <- matrix(runif(200 * 20, 1, 50), ncol = 20)
# the target feature is the last column of the dataset as a vector
y <- rbinom(200, 10, 0.6)
N <- sample(11:20, 200, replace = TRUE)
target <- cbind(y, N)
testIndBinom(target, dataset, xIndex = 14, csIndex = 10)

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

Conditional independence tests for survival data

Description

The main task of this test is to provide a p-value PV_VALUE 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,
         univariateModels = NULL, hash = FALSE, stat_hash = NULL, pvalue_hash = NULL)

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

censIndER(target, dataset, xIndex, csIndex, wei = NULL,
          univariateModels = NULL, hash = FALSE, stat_hash = NULL, pvalue_hash = NULL)
Conditional independence tests for survival data

censIndLLR(target, dataset, xIndex, csIndex, wei = NULL,
univariateModels = NULL, hash = FALSE, stat_hash = NULL, pvalue_hash = NULL)

permCR(target, dataset, xIndex, csIndex, wei = NULL,
univariateModels = NULL, hash = FALSE, stat_hash = NULL, pvalue_hash = NULL,
threshold = 0.05, R = 999)

permWR(target, dataset, xIndex, csIndex, wei = NULL,
univariateModels = NULL, hash = FALSE, stat_hash = NULL, pvalue_hash = NULL,
threshold = 0.05, R = 999)

permER(target, dataset, xIndex, csIndex, wei = NULL,
univariateModels = NULL, hash = FALSE, stat_hash = NULL, pvalue_hash = NULL,
threshold = 0.05, R = 999)

permLLR(target, dataset, xIndex, csIndex, wei = NULL,
univariateModels = NULL, hash = FALSE, stat_hash = NULL, pvalue_hash = NULL,
threshold = 0.05, R = 999)

waldCR(target, dataset, xIndex, csIndex, wei = NULL,
univariateModels = NULL, hash = FALSE, stat_hash = NULL, pvalue_hash = NULL)

waldWR(target, dataset, xIndex, csIndex, wei = NULL,
univariateModels = NULL, hash = FALSE, stat_hash = NULL, pvalue_hash = NULL)

waldER(target, dataset, xIndex, csIndex, wei = NULL,
univariateModels = NULL, hash = FALSE, stat_hash = NULL, pvalue_hash = NULL)

Arguments

target A Survival object (class Surv from package survival) containing the time to event data (time) and the status indicator vector (event). View Surv documentation for more information.
dataset A numeric matrix or data frame, in case of categorical predictors (factors), containing the variables for performing the test. Rows as samples and columns as features. In the cases of "waldCR", "waldWR", "waldER", "waldCR", "permWR", "permER" and "permLLR" this is strictly a matrix.
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. An example where weights are used is surveys when stratified sampling has occured.
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 which contains the cached generated statistics of a SES run in the current dataset, using the current test.

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

threshold Threshold (suitable values in (0, 1)) for assessing p-values significance.

R The number of permutations, set to 999 by default. There is a trick to avoid doing all permutations. As soon as the number of times the permuted test statistic is more than the observed test statistic is more than 50 (in this example case), the p-value has exceeded the significance level (threshold value) and hence the predictor variable is not significant. There is no need to continue do the extra permutations, as a decision has already been made.

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). Note: When there are observations with zero values (time=0) the Weibull and Exponential regressions will not work. Only Cox regression will run. The censIndLLR is the log-logistic regression. This is a pure AFT model, unlike Weibull which is either a proportional hazards model or AFT.

If hash = TRUE, censIndCR, censIndWR, censIndER and censIndLLR 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".

The log-likelihood ratio test used in "censIndCR", "censIndWR" and "censIndER" requires the fitting of two models. The Wald tests used in "waldCR", "waldWR" and "waldER" requires fitting of only one model, the full one. The significance of the variable is examined only. Only continuous (or binary) predictor variables are currently accepted in this test.

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.

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

See Also
SES, censIndWR, testIndFisher, gSquare, testIndLogistic, Surv, anova, CondIndTests

Examples

```r
# create a survival simulated dataset
dataset <- matrix(runif(400 * 20, 1, 100), nrow = 400, ncol = 20)
dataset <- as.data.frame(dataset);
timeToEvent <- numeric(400)
event <- numeric(400)
ca <- numeric(400)
for(i in 1:400) {
timeToEvent[i] <- dataset[i, 1] + 0.5 * dataset[i, 10] + 2 * dataset[i, 15] + runif(1, 0, 3);
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
censIndCR( target, dataset, xIndex = 12, csIndex = c(5, 7, 4) )
# run the SESC algorithm
## Not run:
```
Conditional independence tests with and without permutation p-value

The main task of this test is to provide a permutation based p-value PV ALUE 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 = FALSE, R = 499)
cat.ci(ind1, ind2, cs, dat, type, rob = FALSE, R = 1)

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). If you have no vari-
able set this equal to 0.
dat A matrix with the data. In the case of "cat.ci" the minimum must be 0, i.e. the
data must be like 0, 1, 2 and NOT 1, 2, 3... There is a C++ code behind and the
minimum must be 0.
type Do you want the Pearson (type = "pearson") or the Spearman (type = "spear-
man") 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.
For "cat.ci" this should be a vector with the levels, the number of distinct (dif-
ferent) values of each categorical variable. Its length is equal to the number of
variables used in the test (2 + the number of conditioning variables).
rob If you choose type="pearson" then you can sapecify whether you want a robust
version of it. For "dist.condi" and "cat.ci" 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@uoc.gr>

References


See Also

testIndFisher, testIndSpearman, pc.skel, gSquare, CondIndTests

Examples

#simulate a dataset with continuous data
dataset <- matrix(runif(500 * 5, 1, 100), ncol = 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)
**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**

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

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

wald.ses(target, dataset, max_k = 3, threshold = 0.05, test = NULL, ini = NULL, wei = NULL, user_test = NULL, hash = FALSE, hashObject = NULL, ncores = 1, backward = FALSE)

wald.mmpc(target, dataset, max_k = 3, threshold = 0.05, test = NULL, ini = NULL, wei = NULL, user_test = NULL, hash = FALSE, hashObject = NULL, ncores = 1, backward = FALSE)

perm.ses(target, dataset, max_k = 3, threshold = 0.05, test = NULL, ini = NULL, wei = NULL, user_test = NULL, hash = FALSE, hashObject = NULL, R = 999, ncores = 1, backward = FALSE)

perm.mmpc(target, dataset, max_k = 3, threshold = 0.05, test = NULL, ini = NULL, wei = NULL, user_test = NULL, hash = FALSE, hashObject = NULL, R = 999, 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).
max_k  The maximum conditioning set to use in the conditional independence 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 MMPC) again, you can extract them from the first run of SES and plug them here. This can speed up the second run (and subsequent 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. An example where weights are used is surveys when stratified sampling has occurred.
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.
R  The number of permutations, set to 999 by default. There is a trick to avoid doing all permutations. As soon as the number of times the permuted test statistic is more than the observed test statistic is more than 50 (if threshold = 0.05 and R = 999), the p-value has exceeded the significance level (threshold value) and hence the predictor variable is not significant. There is no need to continue do the extra permutations, as a decision has already been made.
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. It calls the link{mmpcbackphase} for this purpose.
Details

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

The MMPC function implements the MMPC algorithm as presented in "Tsamardinos, Brown and Aliferis. The max-min hill-climbing Bayesian network structure learning algorithm".

The output value "univ" along with the output value "hashObject" can speed up the computations of subsequent 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 natural 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 the unconditional correlation coefficients are calculated very efficiently.s.

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 say 3, 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 \texttt{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 \texttt{testIndSpearman} (argument "test"), the ranks of the data calculated and those are used in the calculations. 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.

\begin{verbatim}
# 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 simulation 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 regression will be flat. If the data come from a negative binomial, the histogram with a Poisson will not be uniform.

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

\end{verbatim}

Value

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

\begin{itemize}
  \item \texttt{selectedVars} The selected variables, i.e., the signature of the target variable.
  \item \texttt{selectedVarsOrder} The order of the selected variables according to increasing p-values.
  \item \texttt{queues} A list containing a list (queue) of equivalent features for each variable included in \texttt{selectedVars}. An equivalent signature can be built by selecting a single feature from each queue. Featured only in SES.
  \item \texttt{signatures} A matrix reporting all equivalent signatures (one signature for each row). Featured only in SES.
  \item \texttt{hashObject} The hashObject caching the statistic calculated in the current run.
  \item \texttt{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 variable. Particularly, this vector reports the max p-values found when the association of each variable
with the target is tested against different conditional sets. Lower values indicate higher association. **Note that these are the logged p-values, natural logarithm of the p-values, and not the p-values.**

*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. 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 subsequent 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.

*n.tests*  
If you have set hash = TRUE, then the number of tests performed by SES or MMPC will be returned. If you have not set this to TRUE, the number of univariate associations will be returned. So be careful with this number.

*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.

**Generic Functions implemented for 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:

- **quantreg**: for the quantile (median) regression
- **MASS**: for negative binomial regression and simple ordinal regression
- **nnet**: also require(stats) and require(MASS) for the testIndLogistic test
- **survival**: for the censIndCR, censIndWR and the censIndER tests
- **doParallel**: for parallel computations
- **lme4**: for (generalised) linear mixed models
- **Rfast**: for many fast functions.

**Author(s)**

Ioannis Tsamardinos, Vincenzo Lagani

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


See Also

CondIndTests, cv.ses

Examples

```r
set.seed(123)

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

define a simulated class variable
target <- 3 * dataset[, 10] + 2 * dataset[, 15] + 3 * dataset[, 20] + rnorm(100, 0, 5)

#define some simulated equivalences
dataset[, 16] <- dataset[, 10] + rnorm(100, 0, 2)
dataset[, 17] <- dataset[, 15] + rnorm(100, 0, 2)

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

# 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);
```
# get the run time
sesObject@runtime;
sesObject2@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.glmm/SES.gee:** Feature selection algorithm for identifying multiple minimal, statistically-equivalent and equally-predictive feature signatures with correlated data

---

**Description**

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

**Usage**

```r
SES.glmm(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.glmm(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.gee(target, reps = NULL, group, dataset, max_k = 3, threshold = 0.05,
test = NULL, ini = NULL, wei = NULL, user_test = NULL, hash = FALSE,
hashObject = NULL, correl = "exchangeable", se = "jack", ncores = 1)

SES.gee(target, reps = NULL, group, dataset, max_k = 3, threshold = 0.05,
test = NULL, ini = NULL, wei = NULL, user_test = NULL, hash = FALSE,
hashObject = NULL, correl = "exchangeable", se = "jack", 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. Its length is equal to the length of the target variable. If you have clustered data, leave this NULL.
Constraint based feature selection algorithms for longitudinal and clustered data

**group**
A numeric vector containing the subjects or groups. It must be of the same length as target.

**dataset**
The dataset; provide either a data frame or a matrix (columns = variables, rows = samples). Currently, only continuous datasets are supported.

**max_k**
The maximum conditioning set to use in the conditional independence 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 tests are the "testIndGLMMLogistic", "testIndGLMMPois", "testIndGLMMGamma", "testIndGLMMNormLog", "testIndGLMMOrdinal", "testIndGLMMReg", "testIndLMM" and "testIndGLMMCR" for generalised linear mixed models. For the GEE, the available tests are "testIndGEEReg", "testIndGEEPois", "testIndGEELogistic", "testIndGEEGamma" and "testIndGEENormLog".

**ini**
This is 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 MMPC) again, you can extract them from the first run of SES and plug them here. This can speed up the second run (and subsequent 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.glmm. 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.

**correl**
The correlation structure. For the Gaussian, Logistic, Poisson and Gamma regression this can be either "exchangeable" (compound symmetry, suitable for clustered data) or "ar1" (AR(1) model, suitable for longitudinal data). For the ordinal logistic regression its only the "exchangeable" correlation structure.

**se**
The method for estimating standard errors. This is very important and crucial. The available options for Gaussian, Logistic, Poisson and Gamma regression are: a) 'san.se': the usual robust estimate. b) 'jack': approximate jackknife variance estimate. c) 'j1s': if 1-step jackknife variance estimate and d) 'fij': fully iterated jackknife variance estimate. If you have many clusters (sets of repeated
measurements) "san.se" is fine as it is asymptotically correct, plus jackknife estimates will take longer. If you have a few clusters, then maybe it’s better to use jackknife estimates.

The jackknife variance estimator was suggested by Paik (1988), which is quite suitable for cases when the number of subjects is small (K < 30), as in many biological studies. The simulation studies conducted by Ziegler et al. (2000) and Yan and Fine (2004) showed that the approximate jackknife estimates are in many cases in good agreement with the fully iterated ones.

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 definitively not linear in the number of cores.

Details

The SES.glmm function implements the Statistically Equivalent Signature (SES) algorithm as presented in "Tsamardinos, Lagani and Pappas, HSCBB 2012" adapted to longitudinal data. The citation for this is "Tsagris, Lagani and tsamardinos, (2018)". These functions presented here are for the temporal-lonitudinal scenario.

The MMPC function implements the MMPC algorithm as presented in "Tsamardinos, Brown and Aliferis. The max-min hill-climbing Bayesian network structure learning algorithm" adapted to longitudinal data.

The output value "univ" along with the output value "hashObject" can speed up the computations of subsequent 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 testIndGLMMReg.

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.

If you want to use the GEE methodology, make sure you load the library geepack first.

Value

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

- **selectedVars**: The selected variables, i.e., the signature of the target variable.
- **selectedVarsOrder**: The order of the selected variables according to increasing p-values.
- **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 found when the association of each variable with the target is tested against different conditional sets. Lower values indicate higher association. **Note that these are the logarithm of the p-values.**
- **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**. 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 subsequent 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.

n.tests
If you have set hash = TRUE, then the number of tests performed by SES or MMPC will be returned. If you have not set this to TRUE, the number of univariate associations will be returned. So be careful with this number.

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.

slope
Whether random slopes for the time effects were used or not, TRUE or FALSE.

Generic Functions implemented for SESoutput Object:

`plot(object=SES.glmm.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.

Author(s)
Ioannis Tsamardinos, Vincenzo Lagani

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

References


See Also

CondIndTests, testIndGLMMReg

Examples

```r
## Not run:
require(lme4)
data(sleepstudy)
reaction <- sleepstudy$Reaction
days <- sleepstudy$Days
subject <- sleepstudy$Subject
x <- matrix(rnorm(180 * 200),ncol = 200) ## unrelated predictor variables
m1 <- SES.glmm(target = reaction, reps = days, group = subject, dataset = x)
m2 <- MMPC.glmm(target = reaction, reps = days, group = subject, dataset = x)
## End(Not run)
```

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

```r
ma.ses(target, dataset, ina, statistic = FALSE, max_k = 3, threshold = 0.05, 
       test = NULL, ini = NULL, user_test = NULL, hash = FALSE, hashObject = NULL, 
       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, 
        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 independence 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 subsequent 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.

**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. This is an experimental stage, so do not trust too much or ven better do not use.
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".

The MMPC function implements the MMPC algorithm as presented in "Tsamardinos, Brown and Aliferis. The max-min hill-climbing Bayesian network structure learning algorithm".

The output value "univ" along with the output value "hashObject" can speed up the computations of subsequent 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 the unconditional correlation coefficients are calculated very efficiently.

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 calculations. 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 $T \chi^2_2$. If statistic is TRUE, the test statistics are combined as $T = \frac{\sum_{i=1}^k t_i/\sigma(t_i)}{\sum_{i=1}^k 1/\sigma(t_i)}$ and $T \sim N(0, 1)$. 

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 (on a logarithmic scale) found 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 performed 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 subsequent 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.

Generic Function implemented for 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.
Constraint based feature selection algorithms for multiple datasets

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@uoc.gr>

References


See Also

SES, CondIndTests, cv.ses

Examples

set.seed(123)

#simulate a dataset with continuous data
dataset <- matrix(runif(400 * 100, 1, 100), ncol = 100)

#define a simulated class variable
target <- 3 * dataset[, 10] + 2 * dataset[, 20] + 3 * dataset[, 30] + rnorm(400, 0, 5)

#define some simulated equivalences
dataset[, 15] <- dataset[, 10] + rnorm(400, 0, 2)
dataset[, 10] <- dataset[, 10] + rnorm(400, 0, 2)
dataset[, 25] <- dataset[, 20] + rnorm(400, 0, 2)
dataset[, 23] <- dataset[, 20] + rnorm(400, 0, 2)

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

\[
\text{testIndFisher}(\text{target}, \text{dataset}, \text{xIndex}, \text{csIndex}, \text{wei} = \text{NULL}, \text{statistic} = \text{FALSE}, \text{univariateModels} = \text{NULL}, \text{hash} = \text{FALSE}, \text{stat_hash} = \text{NULL}, \text{pvalue_hash} = \text{NULL})
\]

\[
\text{testIndMFFisher}(\text{target}, \text{dataset}, \text{xIndex}, \text{csIndex}, \text{wei} = \text{NULL}, \text{statistic} = \text{FALSE}, \text{univariateModels} = \text{NULL}, \text{hash} = \text{FALSE}, \text{stat_hash} = \text{NULL}, \text{pvalue_hash} = \text{NULL})
\]

\[
\text{testIndSpearman}(\text{target}, \text{dataset}, \text{xIndex}, \text{csIndex}, \text{wei} = \text{NULL}, \text{statistic} = \text{FALSE}, \text{univariateModels} = \text{NULL}, \text{hash} = \text{FALSE}, \text{stat_hash} = \text{NULL}, \text{pvalue_hash} = \text{NULL})
\]

\[
\text{permFisher}(\text{target}, \text{dataset}, \text{xIndex}, \text{csIndex}, \text{wei} = \text{NULL}, \text{statistic} = \text{FALSE}, \text{univariateModels} = \text{NULL}, \text{hash} = \text{FALSE}, \text{stat_hash} = \text{NULL}, \text{pvalue_hash} = \text{NULL}, \text{threshold} = 0.05, \text{R} = 999)
\]

\[
\text{permMFFisher}(\text{target}, \text{dataset}, \text{xIndex}, \text{csIndex}, \text{wei} = \text{NULL}, \text{statistic} = \text{FALSE}, \text{univariateModels} = \text{NULL}, \text{hash} = \text{FALSE}, \text{stat_hash} = \text{NULL}, \text{pvalue_hash} = \text{NULL}, \text{threshold} = 0.05, \text{R} = 999)
\]

\[
\text{permDcor}(\text{target}, \text{dataset}, \text{xIndex}, \text{csIndex}, \text{wei} = \text{NULL}, \text{statistic} = \text{FALSE}, \text{univariateModels} = \text{NULL}, \text{hash} = \text{FALSE}, \text{stat_hash} = \text{NULL}, \text{pvalue_hash} = \text{NULL}, \text{threshold} = 0.05, \text{R} = 499)
\]

Arguments

\text{target} \hspace{1cm} 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 \( \log(\text{target}/(1 - \text{target})) \). This can also be a list of vectors as well. In this case, the metanalytic approach is used.

\text{dataset} \hspace{1cm} A numeric matrix containing the variables for performing the test. Rows as samples and columns as features.

\text{xIndex} \hspace{1cm} The index of the variable whose association with the target we want to test.
Correlation based conditional independence tests

- **csIndex**: The indices of the variables to condition on. If you have no variables set this equal to 0.

- **wei**: A vector of weights to be used for weighted regression. The default value is NULL. This is not used with "permDcor".

- **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.

- **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 successful) 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 which contains the cached generated statistics of a SES run in the current dataset, using the current test.

- **pvalue_hash**: A hash object which contains the cached generated p-values of a SES run in the current dataset, using the current test.

- **threshold**: Threshold (suitable values in (0, 1)) for assessing p-values significance. Default value is 0.05. This is actually obsolete here, but has to be in order to have a concise list of input arguments across the same family of functions.

- **R**: The number of permutations to use. The default value is 999. For the "permDcor" this is set to 499.

**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 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. It 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 and permDcor tests. The permDcor test uses the distance correlation instead of the usual Pearson or Spearman correlations.

TestIndMMFisher does a robust estimation of the correlation via MM regression.

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).
- **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


See Also
testIndSpearman, testIndReg, SES, testIndLogistic, gSquare, CondIndTests

Examples

```r
#simulate a dataset with continuous data
dataset <- matrix(runif(300 * 50, 1, 1000), nrow = 50 )
#the target feature is the last column of the dataset as a vector
target <- dataset[, 50]
res1 <- testIndFisher(target, dataset, xIndex = 44, csIndex = 10)
res2 <- testIndSpearman(target, dataset, xIndex = 44, csIndex = 10)
res3 <- permFisher(target, dataset, xIndex = 44, csIndex = 10, R = 999)
res4 <- permDcor(target, dataset, xIndex = 44, csIndex = 10, R = 99)

#define class variable (here the last column of the dataset)
dataset <- dataset[, -50]
#run the MMPC algorithm using the testIndFisher conditional independence test
mmpcObject <- MMPC(target, dataset, max_k = 3, threshold = 0.05, test = "testIndFisher")
```

Cross-Validation for gOMP

Description

The function performs a k-fold cross-validation for identifying the best tolerance values for gOMP.

Usage

```r
cv.gomp(target, dataset, kfolds = 10, folds = NULL, tol = seq(4, 9, by = 1),
task = "C", metric = NULL, metricbbc = NULL, modeler = NULL, test = NULL,
method = "ar2", B = 1)
```
Arguments

target
The target or class variable as in SES and MMPC. The difference is that it cannot accept a single numeric value, an integer indicating the column in the dataset.
dataset
The dataset object as in SES and MMPC.
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.
tol
A vector of tolerance values.
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 regression (Cox, Weibull or exponential regression).
mimetic
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 choosing something else. Note that you put these words as they are, without "".
mimetricbbc
This is the same argument as "metric" with the difference that " " must be placed. If for example, metric = auc.mxm, here metricbbc = "auc.mxm". The same value must be given here. This argument is to be used with the function bbc which does bootstrap bias correction of the estimated performance (Tsamardinos, Greasidou and Borboudakis, 2018). This argument is valid if the last argument (B) is more than 1.
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 choosing something else. Note that you put these words as they are, without "".
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 "censIndCR" are used for classification, regression and survival analysis tasks, respectively. If you know what you have put it here to avoid the function choosing something else. Not all tests can be included here. "testIndClogit", "testIndMVreg", "testIndIG", "testIndGamma", "testIndZIP" and "testIndTobit" are not available at the moment.
method
This is only for the "testIndFisher". You can either specify, "ar2" for the adjusted R-square or "sse" for the sum of squares of errors. The tolerance value in both cases must a number between 0 and 1. That will denote a percentage. If the percentage increase or decrease is less than the numerator the algorithm stops. An alternative is "BIC" for BIC and the tolerance values are like in all other regression models.
B
How many bootstrap re-samples to draw. This argument is to be used with the function bbc which does bootstrap bias correction of the estimated performance (Tsamardinos, Greasidou and Borboudakis, 2018). If you have thousands of
samples (observations) then this might not be necessary, as there is no optimistic bias to be corrected. What is the lower limit cannot be told beforehand however. SES and MMPC however were designed for the low sample cases, hence, bootstrap bias correction is perhaps a must thing to do.

**Details**

For more details see also `cv.ses`.

**Value**

A list including:

- `cv_results_all` A list with predictions, performances and selected variables for each fold and each tolerance value. The elements are called "preds", "performances" and "selectedVars".
- `best_performance` A numeric value that represents the best average performance.
- `best_configuration` A numeric value that represents the best tolerance value.
- `bbc_best_performance` The bootstrap bias corrected best performance if B was more than 1, otherwise this is NULL.
- `runtime` The runtime of the cross-validation procedure.

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: Michail Tsagris <mtsagris@uoc.gr>.

**References**


**See Also**

`cv.mmpc`, `gomp.path`, `bbc`
Examples

```r
## Not run:
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]
# run a 10 fold CV for the regression task
best_model <- cv.gomp(target, dataset, kfolds = 5, task = "R",
tol = seq(0.001, 0.01, by=0.001), method = "ar2" )
## End(Not run)
```

Cross-validation for ridge regression

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 a grid of values of $\lambda$ 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 \( \lambda \), then specify `auto = TRUE` and the \( \lambda \) which minimizes the generalised cross-validation criterion will be returned. Otherwise 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 \( \lambda \).
- `lambda`: If auto is FALSE the \( \lambda \) 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@uoc.gr>

References


See Also

`ridge.reg`

Examples

```r
# simulate a dataset with continuous data
dataset <- matrix(runif(200 * 40, 1, 100), nrow = 200)
# the target feature is the last column of the dataset as a vector
target <- dataset[, 40]
a1 <- ridgereg.cv(target, dataset, auto = TRUE)
a2 <- ridgereg.cv( target, dataset, K = 10, lambda = seq(0, 1, by = 0.1) )
```
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, metricbbc = NULL, modeler = NULL, ses_test = NULL, ncores = 1, B = 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, metricbbc = NULL, modeler = NULL, mmpc_test = NULL, ncores = 1, B = 1)

cv.waldses(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, metricbbc = NULL, modeler = NULL, ses_test = NULL, ncores = 1, B = 1)

cv.waldmmpc(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, metricbbc = NULL, modeler = NULL, mmpc_test = NULL, ncores = 1, B = 1)

cv.permses(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, metricbbc = NULL, modeler = NULL, ses_test = NULL, R = 999, ncores = 1, B = 1)

cv.permmmpc(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, metricbbc = NULL, modeler = NULL, mmpc_test = NULL, R = 999, ncores = 1, B = 1)

Arguments

- **target**: The target or class variable as in SES and MMPC. The difference is that it cannot accept a single numeric value, an integer indicating the column in the dataset.

- **dataset**: The dataset object as in SES and MMPC.
Cross-Validation for 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 choosing something else. Note that you put these words as they are, without "".

metricbbc
This is the same argument as "metric" with the difference that " " must be placed. If for example, metric = auc.mxm, here metricbbc = "auc.mxm". The same value must be given here. This argument is to be used with the function bbc which does bootstrap bias correction of the estimated performance (Tsamardinos, Greasidou and Borboudakis, 2018). This argument is valid if the last argument (B) is more than 1.

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 choosing something 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 "censIndCR" are used for classification, regression and survival analysis tasks, respectively. If you know what you have put it here to avoid the function choosing something else. Not all tests can be included here. "testIndClogit", "testIndMVreg", "testIndIG", "testIndGamma", "testIndZIP" and "testIndTobit" are not available at the moment.

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 "censIndCR" are used for classification, regression and survival analysis tasks, respectively.

R
The number of permutations, set to 999 by default. There is a trick to avoid doing all permutations. As soon as the number of times the permuted test statistic is more than the observed test statistic is more than 50 (if threshold = 0.05 and R = 999), the p-value has exceeded the signifiance level (threshold value) and hence the predictor variable is not significant. There is no need to continue do the extra permutations, as a decision has already been made.
**ncores**  
This argument is valid only if you have a multi-threaded machine.

**B**  
How many bootstrap re-samples to draw. This argument is to be used with the function `bbc` which does bootstrap bias correction of the estimated performance (Tsamardinos, Greasidou and Borboudakis, 2018). If you have thousands of samples (observations) then this might not be necessary, as there is no optimistic bias to be corrected. What is the lower limit cannot be told beforehand however. SES and MMPC however were designed for the low sample cases, hence, bootstrap bias correction is perhaps a must thing to do.

**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 for binary logistic regression.
- **acc.mxm:** accuracy for binary logistic regression.
- **fscore.mxm:** F score for binary logistic regression.
- **prec.mxm:** precision for binary logistic regression.
- **euclid_sens.spec.mxm:** Euclidean norm of 1 - sensitivty and 1 - specificity for binary logistic regression.
- **spec.mxm:** specificity for logistic regression.
- **sens.mxm:** sensitivity for logistic regression.
- **acc.multinom.mxm:** accuracy for multinomial logistic regression.
- **mse.mxm:** mean squared error, for robust and non robust linear regression and median (quantile) regression (multiplied by -1).
- **pve.mxm:** 1 - (mean squared error)/( (n - 1) * var(y_out) ), for non robust linear regression. It is basically the proportion of variance explained in the test set.
- **ci.mxm:** 1 - concordance index as provided in the rcorr.cens function from the suviva 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 survival package. This is to be used with the Weibull regression model only.
- **poisdev.mxm:** Poisson regression deviance.
- **nbdev.mxm:** Negative binomial regression deviance (multiplied by -1).
- **binomdev.mxm:** Negative binomial regression deviance (multiplied by -1).
- **ord_mae.mxm:** Ordinal regression mean absolute error (multiplied by -1).
- **mae.mxm:** Mean absolute error (multiplied by -1).
- **mci.mxm:** Matched concordance index (for conditional logistic regression).
Usage: metric(predictions, test_target)


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).
- multinom.mxm: fits a multinomial regression model (classification task).
- lm.mxm: fits a linear model (regression task).
- coxph.mxm: fits a cox proportional hazards regression model (survival task).
- weibreg.mxm: fits a Weibull regression model (survival task).
- rq.mxm: fits a quantile (median) regression model (regression task).
- lmrob.mxm: fits a robust linear model (regression task).
- pois.mxm: fits a poisson regression model (regression task).
- nb.mxm: fits a negative binomial regression model (regression task).
- ordinal.mxm: fits an ordinal regression model (regression task).
- beta.mxm: fits a beta regression model (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.
- clogit: fits a conditional logistic regression model.

Usage: modeler(train_target, sign_data, sign_test)

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.
- best_configuration: A list that corresponds to the best configuration of SES or MMPC including id, threshold (named 'a') and max_k.
The bootstrap bias corrected best performance if B was more than 1, otherwise this is NULL.

The runtime of the cross-validation procedure.

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


See Also

SES, CondIndTests, cv.gomp, bbc, testIndFisher, testIndLogistic, gSquare, censIndCR

Examples

```r
set.seed(1234)

# simulate a dataset with continuous data
dataset <- matrix(rnorm(100 * 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]

# 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
```
Cross-validation of the FBED with LMM

best_model$best_performance

# summary elements of the process. Press tab after each $ to view all the elements and
# choose the one you are interesting 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

Cross-validation of the FBED with LMM

Cross-validation of the FBED with LMM

Description

Cross-validation of the FBED with LMM.

Usage

cv.fbed.lmm.reg(target, dataset, id, prior = NULL, kfolds = 10,
folds = NULL, alphas = c(0.01, 0.05), ks = 0:2)

Arguments

target The class variable. This must be a numerical vector with continuous data.
dataset The dataset: provide a numerical a matrix (columns = variables, rows = observations).
id This is a numerical vector of the same size as target denoting the groups or the subjects.
prior If you have prior knowledge of some variables that must be in the variable selection phase add them here. This an be a vector (if you have one variable) or a matrix (if you more variables). This does not work during the backward phase at the moment.
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 significance levels to be tested.
ks A vector of K values to be tested.

Details

The function performs cross-validation for the FBED algorithm with clustered data using the linear mixed model. The k-folds cross-validation is on clusters. Instead of leaving observations, clusters are left aside each time.
Cross-validation of the FBED with LMM

Value

A list including:

list(vars = vars, cv = cv, perf = perf, best = best, runtime = runtime)

vars  
An array with the number of selected variables for each combination of significance level and value of K.

CV  
An array with the number of selected variables for each combination of significance level and value of K.

perf  
A matrix with the average performance each combination of significance level and value of K.

best  
The best significance level and value of K.

runtime  
The runtime required.

Author(s)

Michail Tsagris

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

References


See Also

fbed.glmm.reg, fbed.gee.reg, MMPC.glmm

Examples

```r
## Not run:
require(lme4)
data(sleepstudy)
reaction <- sleepstudy$Reaction
subject <- sleepstudy$Subject
x1 <- sleepstudy$Days
x <- matrix(rnorm(180 * 200), ncol = 200) # unrelated predictor variables
x <- cbind(x1, x)
m <- cv.fbed.lmm.reg(reaction, x, subject)
## End(Not run)
```
Data simulation from a DAG

Data simulation from a DAG

Description

Data simulation from a DAG.

Usage

\[ \text{rdag}(n, p, s, a = 0, m, A = \text{NULL}, \text{seed} = \text{FALSE}) \]
\[ \text{rdag2}(n, A = \text{NULL}, p, \text{nei}, \text{low} = 0.1, \text{up} = 1) \]
\[ \text{rmdag}(n, A = \text{NULL}, p, \text{nei}, \text{low} = 0.1, \text{up} = 1) \]

Arguments

\begin{itemize}
  \item \textbf{n} \hspace{1cm} A number indicating the sample size.
  \item \textbf{p} \hspace{1cm} A number indicating the number of nodes (or vertices, or variables).
  \item \textbf{nei} \hspace{1cm} The average number of neighbours.
  \item \textbf{s} \hspace{1cm} A number in \((0, 1)\). This defines somehow the sparseness of the model. It is the probability that a node has an edge.
  \item \textbf{a} \hspace{1cm} 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.
  \item \textbf{m} \hspace{1cm} 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 mean vector of the multivariate normal from which the outliers will be generated.
  \item \textbf{A} \hspace{1cm} If you already have an an adjacency matrix in mind, plug it in here, otherwise, leave it \text{NULL}.
  \item \textbf{seed} \hspace{1cm} If seed is \text{TRUE}, the simulated data will always be the same.
  \item \textbf{low} \hspace{1cm} Every child will be a function of some parents. The beta coefficients of the parents will be drawn uniformly from two numbers, low and up. See details for more information on this.
  \item \textbf{up} \hspace{1cm} Every child will be a function of some parents. The beta coefficients of the parents will be drawn uniformly from two numbers, low and up. See details for more information on this.
\end{itemize}

Details

In the case where no adjacency matrix is given, an \(p \times p\) matrix with zeros everywhere is created. Every 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 \([0, 1]\). 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 \((I_p - A)^{-1} (I_p - A)\), where \(I_p\)
is the $p \times p$ identity 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 can specify 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.

For the "rdag2", this is a different way of simulating data from DAGs. The first variable is normally generated. Every other variable can be a function of some previous ones. Suppose now that the $i$-th variable is a child of 4 previous variables. We need for coefficients $b_j$ to multiply the 4 variables and then generate the $i$-th variable from a normal with mean $\sum_{j=1}^4 b_j X_j$ and variance 1. The $b_j$ will be either positive or negative values with equal probability. Their absolute values ranges between "low" and "up". The code is accessible and you can see in detail what is going on. In addition, every generated data, are standardised to avoid numerical overflow.

The "rmdag" generates data from a BN with continuous, ordinal and binary data in proportions 50%, 25% and 25% respectively on average. This was used in the experiments run by Tsagris et al. (2017). If you want to generate data and then use them in the "pcalg" package with the function "ci.fast2" or "ci.mm2" you should transform the resulting data into a matrix. The factor variables must become numeric starting from 0. See the examples for more on this.

### Value

- `nout`: The number of outliers.
- `G`: The adjacency matrix used. For the "rdag" if $G[i, j] = 2$, then $G[j, i] = 3$ and this means that there is an arrow from $j$ to $i$. For the "rdag2" and "rmdag" the entries are either $G[i, j] = G[j, i] = 0$ (no edge) or $G[i, j] = 1$ and $G[j, i] = 0$ (indicating $i \rightarrow j$).
- `A`: The matrix with the with the uniform values in the interval $0.1, 1$. This is returned only by "rdag".
- `x`: The simulated data.

### Author(s)

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

### References


Drop all possible single terms from a model using the partial correlation

See Also

`pc.skel, pc.or, ci.mm, mmhc.skel`

Examples

```r
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)
```

Description

Drop all possible single terms from a model using the partial correlation.

Usage

```r
cor.drop1(y, x, logged = FALSE)
```

Arguments

- `y`: A numerical vector with the response variable.
- `x`: A numerical matrix or a data.frame with the predictor variables. If is a matrix it is internally transformed into a data.frame form, hence the user is advised to supply a data.frame in order to save some time. If the number of columns (variables) is higher than the number of rows (observations) the function will simply not work.
- `logged`: If you want the p-values be returned leave this FALSE. If it is TRUE their logarithm is returned.

Details

This uses R’s command `drop1` and modifies it so as to calculate the p-value using Fisher’s conditional independence test.

Value

A matrix with two columns, the test statistic values and its associated p-value.
**Author(s)**
Michail Tsagris
R implementation and documentation: Michail Tsagris <mtsagris@uoc.gr>

**See Also**
glm.bsreg, fbed.reg, mmpcbackphase

**Examples**
```r
y <- rnorm(200)
x <- matrix( rnorm(200 * 10), ncol = 10)
cor.drop1(y, x)
```

**Description**
eBIC for many regression models.

**Usage**
```r
ebic.regs(target, dataset, xIndex, csIndex, gam = NULL, test = NULL, wei = NULL, ncores = 1)
```

**Arguments**
- `target`: The target (dependent) variable. It must be a numerical vector, a factor or a Surv object.
- `dataset`: The independent variable(s). This can be a matrix or a dataframe with continuous only variables, a data frame with mixed or only categorical variables.
- `xIndex`: The indices of the variables whose association with the target you want to test.
- `csIndex`: The index or indices of the variable(s) to condition on. If this is 0, the function `univregs` will be called.
- `gam`: In case the method is chosen to be "eBIC" one can also specify the `gamma` parameter. The default value is "NULL", so that the value is automatically calculated.
- `test`: One of the following: `testIndBeta`, `testIndReg`, `testIndLogistic`, `testIndOrdinal`, `testIndPois`, `testIndZIP`, `testIndNB`, `testIndClogit`, `testIndBinom`, `testIndClogit`, `censIndCR`, `censIndWR`, `censIndER`, `censIndLLR testIndMultinom`, `testIndTobit`, `testIndSPML`, `testIndGamma` or `testIndNormLog`. Note that in all cases you must give the name of the test, without " ".

**Note**

A vector of weights to be used for weighted regression. The default value is NULL. An example where weights are used is surveys when stratified sampling has occurred.

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 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. The function does not check for zero variance columns, only the "univregs" and related functions do.

Value

A list including:

- `stat`: The value of the test statistic.
- `pvalue`: The logarithm of the p-value of the test.

Author(s)

Michail Tsagris

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

References


See Also

`univregs`, `SES`, `MMPC`, `CondIndTests`, `reg.fit`, `ridge.reg`
Effective sample size for $G^2$ test in BNs with case control data

Examples

```r
y <- rpois(100, 15)
x <- matrix(rnorm(100 * 10), ncol = 10)
a1 <- univregs(y, x, test = testIndPois)
a2 <- perm.univregs(y, x, test = permPois)
a3 <- wald.univregs(y, x, test = waldPois)
a4 <- cond.regs(y, as.data.frame(x), xIndex = 1:4, csIndex = 5, test = testIndPois)
```

Description

Effective sample size for $G^2$ test in BNs with case control data.

Usage

```r
Ness(propNt, N, K = 10000)
```

Arguments

- **propNt**: A numerical vector with the proportions (distribution) of the (single) selection variable.
- **N**: The sample size of the data.
- **K**: The number of repetitions to be used for estimating the effective sample size.

Details

When dealing with case control data, spurious correlations or relationships arise. To deal with this one way is to adjust the sample size used in the $G^2$ test statistic. This function does exactly this, estimates the effective sample size as per the Borboudakis and Tsamardinos (2012) suggestion. The idea is that after learning the skeleton with the usual $G^2$ test, one should go to the edges and perform a conditional $G^2$

Value

The estimated effective sample size.

Author(s)

Michail Tsagris

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

References

Estimation of the percentage of Null p-values

See Also

SES, MMPC, testIndLogistic

Examples

Ness(c(0.3, 0.7), N = 1000, K = 10000)

Description

Estimation of the percentage of Null p-values.

Usage

pi0est(p, lambda = seq(0.05, 0.95, by = 0.01), dof = 3)

Arguments

p
A vector of p-values.

lambda
A vector of values of the tuning parameter lambda.

dof
Number of degrees of freedom to use when estimating pi_0 with smoothing splines.

Details

The estimated portion of null p-values is estimated the algorithm by Storey and Tibshirani (2003).

Value

The estimated proportion of non significant (null) p-values. In the paper Storey and Tibshirani mention that the estimate of pi0 is with lambda=1, but in their R code they use the highest value of lambda and thus we do the same here.

Author(s)

Michail Tsagris

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

References

Fast MMPC

Description

A fast version of MMPC

Usage

```r
mmpc2(target, dataset, prior = NULL, max_k = 3, threshold = 0.05,
      test = "testIndLogistic", ini = NULL, wei = NULL, 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).
- `prior` If you have prior knowledge of some variables that must be in the variable selection phase add them here. This an be a vector (if you have one variable) or a matrix (if you more variables). This does not work during the backward phase at the moment.
- `max_k` The maximum conditioning set to use in the conditional independence 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` One of the following: "testIndBeta", "testIndReg", "testIndRQ", "testIndLogistic", "testIndMultinom", "testIndOrdinal", "testIndPois", "testIndQPois", "testIndZIP", "testIndNB", "testIndClogit", "testIndBinom", "testIndQBinom", "testIndIGreg", "censIndCR", "censIndWR", "censIndER", "testIndMMReg", "testIndMVreg", "testIndMultinom", "testIndOrdinal", "testIndTobit", "testIndGamma", "testIndNormLog" or "testIndSPML".

Examples

```r
## simulate a dataset with continuous data
y <- rdag2(1000, p = 20, nei = 3)
ind <- sample(1:20, 20)
x <- y$x[, ind]
mod <- pc.skel(x, method = "comb.fast", alpha = 0.01)
pval <- exp(mod$pvalue)
pval <- lower.tri(pval)
pi0est(pval)
```
This is a supposed to be a list. To avoid calculating the univariate associations (first step of SES, MMPC and of FBED) again, you can extract them from the first run of one of these algorithms and plug them here. This can speed up the second run (and subsequent runs of course) by 50%. See the details and the argument "univ" in the output values.

A vector of weights to be used for weighted regression. The default value is NULL. An example where weights are used is surveys when stratified sampling has occurred.

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.

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. It calls the \link{mmpcbackphase} for this purpose.

Details

MMPC tests each feature for inclusion (selection). For each feature, it performs a conditional independence test. Each test requires fitting two regression models, one without the feature and one with the feature included. In this version, we have changed the order of the tests. We find all possible subsets of the already selected features and for each of them we test each feature. This way, only half of the regression models the usual MMPC fits, are fitted. Also, less tests will be performed. It is the same algorithm, with a change in the sequence.

We have seen a 50% in the computational time, but the drawback is that if you want to run MMPC with different values of "max$_$k" and "alpha", this is not possible from here. This function is for a single pair of "max$_$k" and "alpha" values. It saves no test statistics, only p-values, no hashing and hence is memory efficient, but contains less information than \code{MMPC}.

Value

The output of the algorithm is an S3 object including:

- \code{selectedVars}: The selected variables, i.e., the signature of the target variable.
- \code{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 found when the association of each variable with the target is tested against different conditional sets. Lower values indicate higher association. \textbf{Note that these are the logarithm of the p-values}
- \code{univ}: A vector with the \textbf{logged p-values} of the univariate associations. This vector is very important for subsequent runs of MMPC with different hyper-parameters. After running SES with some hyper-parameters you might want to run MMPC again with different hyper-parameters. To avoid calculating the univariate
associations (first step) 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 MMPC. This can speed up the second run (and subsequent runs of course) by 50%. See the argument "univ" in the output values.

**kapa_pval**
A list with the same number of elements as the `max_k$. Every element in the list is a matrix. The first row is the **logged p-values**, the second row is the variable whose conditional association with the target variable was tests and the other rows are the conditioning variables.

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

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

**n.tests**
The number of tests performed by MMPC will be returned.

**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.

### Author(s)
Ioannis Tsamardinos, Michail Tsagris

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

### References


### See Also

`MMPC`, `certificate.of.exclusion2`

### Examples

```r
set.seed(123)

# simulate a dataset with continuous data
dataset <- matrix(runif(100 * 40, 1, 100), ncol = 40)

dataset <- matrix(runif(100 * 40, 1, 100), ncol = 40)

target <- 3 * dataset[, 10] + 2 * dataset[, 15] + 3 * dataset[, 20] + rnorm(100, 0, 5)
```
Fast MMPC for longitudinal and clustered data

```r
m <- median(target)
target[target < m] <- 0
target[abs(target) > 0 ] <- 1

m1 <- mmpc2(target, dataset, max_k = 3, threshold = 0.05, test="testIndLogistic")
m1$selectedVars  ## S3 class, $, not @
m1$runtime

m2 <- MMPC(target, dataset, max_k = 3, threshold = 0.05, test="testIndLogistic")
m2@selectedVars # S3 class, @, not $
m2@runtime
```

---

**Fast MMPC for longitudinal and clustered data**

**mmpc.glmm2/mmpc.gee2:** Fast Feature selection algorithm for identifying minimal feature subsets with correlated data

---

**Description**

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

**Usage**

```r
mmpc.glmm2(target, reps = NULL, group, dataset, prior = NULL, max_k = 3,
threshold = 0.05, test = NULL, ini = NULL, wei = NULL, slopes = FALSE, ncores = 1)
mmpc.gee2(target, reps = NULL, group, dataset, prior = NULL, max_k = 3,
threshold = 0.05, test = NULL, ini = NULL, wei = NULL, correl = "exchangeable",
se = "jack", ncores = 1)
```

**Arguments**

- **target**
  - The class variable. Provide a vector with continuous (normal), binary (binomial) or discrete (Poisson) data. For the GEE the data must be sorted so that observations on a cluster are contiguous rows for all entities in the formula.

- **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. For the GEE the data must be sorted so that observations on a cluster are contiguous rows for all entities in the formula.

- **group**
  - A numeric vector containing the subjects or groups. It must be of the same length as target. For the GEE the data must be sorted so that observations on a cluster are contiguous rows for all entities in the formula.
dataset
The dataset; provide either a data frame or a matrix (columns = variables, rows = samples). Currently, only continuous datasets are supported. For the GEE the data must be sorted so that observations on a cluster are contiguous rows for all entities in the formula.

prior
If you have prior knowledge of some variables that must be in the variable selection phase add them here. This can be a vector (if you have one variable) or a matrix (if you more variables).

max_k
The maximum conditioning set to use in the conditional independence 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 are \(\text{testIndGLMMLogistic} \), \(\text{testIndGLMPois} \), \(\text{testIndGLMMGamma} \), \(\text{testIndGLMMNormLog} \) and \(\text{testIndGLMMReg} \) which fit linear mixed models. For the GEE the options are \(\text{testIndGEELogistic} \), \(\text{testIndGEEPois} \), \(\text{testIndGEEGamma} \), \(\text{testIndGEENormLog} \) and \(\text{testIndGEENormLog} \).

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 subsequent 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.

slopes
Should random slopes for the ime effect be fitted as well? Default value is FALSE.

correl
The correlation structure of the GEE. For the Gaussian, Logistic, Poisson and Gamma regression this can be either "exchangeable" (compound symmetry, suitable for clustered data) or "ar1" (AR(1) model, suitable for longitudinal data). All observations must be ordered according to time within each subject. See the vignette of \texttt{geepack} to understand how.

se
The method for estimating standard errors. This is very important and crucial. The available options for Gaussian, Logistic, Poisson and Gamma regression are: a) 'san.se': the usual robust estimate. b) 'jack': approximate jackknife variance estimate. c) 'j1s': if 1-step jackknife variance estimate and d) 'fij': fully iterated jackknife variance estimate. If you have many clusters (sets of repeated measurements) "san.se" is fine as it is asymptotically correct, plus jackknife estimates will take longer. If you have a few clusters, then maybe it’s better to use jackknife estimates.

The jackknife variance estimator was suggested by Paik (1988), which is quite suitable for cases when the number of subjects is small (\(K < 30\)), as in many biological studies. The simulation studies conducted by Ziegler et al. (2000) and Yan and Fine (2004) showed that the approximate jackknife estimates are in many cases in good agreement with the fully iterated ones.

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 definately not linear in the number of cores.

Details

These are faster versions of MMPC using GLMM or GEE methodologies. See also mmpc2 for more details on the algorithm.

If you want to use the GEE methodology, make sure you load the library geepack first.

Value

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

- **selectedVars**: The selected variables, i.e., the signature of the target variable.
- **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 found when the association of each variable with the target is tested against different conditional sets. Lower values indicate higher association. **Note that these are the logarithm of the p-values.**
- **univ**: A vector with the logged p-values of the univariate associations. This vector is very important for subsequent runs of MMPC with different hyper-parameters. After running SES with some hyper-parameters you might want to run MMPC again with different hyper-parameters. To avoid calculating the univariate associations (first step) 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 MMPC. This can speed up the second run (and subsequent runs of course) by 50%. See the argument "univ" in the output values.
- **kapa_pval**: A list with the same number of elements as the max_k. Every element in the list is a matrix. The first row is the logged p-values, the second row is the variable whose conditional association with the target variable was tests and the other rows are the conditioning variables.
- **max_k**: The max_k option used in the current run.
- **threshold**: The threshold option used in the current run.
- **n.tests**: If you have set hash = TRUE, then the number of tests performed by SES or MMPC will be returned. If you have not set this to TRUE, the number of univariate associations will be returned. So be careful with this number.
- **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.
- **slope**: Whether random slopes for the time effects were used or not, TRUE or FALSE.
Fast MMPC for longitudinal and clustered data

Author(s)
Ioannis Tsamardinos, Vincenzo Lagani

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

References


See Also
CondIndTests, testIndGLMMReg

Examples
```r
## Not run:
require(lme4)
data(sleepstudy)
reaction <- sleepstudy$Reaction
days <- sleepstudy$Days
```
subject <- sleepstudy$Subject
x <- matrix(rnorm(180 * 100), ncol = 100) ## unrelated predictor variables
m1 <- mmpc.glmm2(target = reaction, reps = days, group = subject, dataset = x)
m2 <- MMPC.glmm(target = reaction, reps = days, group = subject, dataset = x)

## End(Not run)

Feature selection using SES and MMPC for classification with longitudinal data

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.timeclass(target, reps, id, dataset, max_k = 3, threshold = 0.05,
ini = NULL, wei = NULL, hash = FALSE, hashObject = NULL, ncores = 1)

MMPC.timeclass(target, reps, id, dataset, max_k = 3, threshold = 0.05,
ini = NULL, wei = NULL, hash = FALSE, hashObject = NULL, ncores = 1)
```

Arguments

- `target` The class variable. Provide a vector or a factor with discrete numbers indicating the class. Its length is equal to the number of rows of the dataset.
- `reps` A numeric vector containing the time points of the subjects. Its length is equal to the number of rows of the dataset.
- `id` A numeric vector containing the subjects or groups. Its length is equal to the number of rows of the dataset.
- `dataset` The dataset; provide a matrix. Currently, only continuous datasets are supported. The dataset contains longitudinal data, where each column is a variable. The repeated measurements are the samples.
- `max_k` The maximum conditioning set to use in the conditional independence 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.
- `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 subsequent 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.

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.glmm. 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.

ncores
How many cores to use. This plays an important role if you have tens of thou-
sands 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 definitely not linear in the number of cores.

Details
This is SES and MMPC used in the static-longitudinal scenario of Tsagris, Lagani and Tsamardinos
(2018). The idea is that you have many features of longitudinal data for many subjects. For each
subject you have calculated the coefficients of a simple linear regression over time and this is re-
peated for each feature. In the end, assuming p features, you have p constants and p slopes for each
subject, each constant and slope refers to a feature for a subject. Hence, each new feature consists
of two vectors, the constants and the slopes and the feature selection takes place there.

Value
The output of the algorithm is an object of the class 'SESoutput' for SES or 'MMPConoutput' 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). Fea-
tured 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 found when the association of each variable with the target is tested against different conditional sets. Lower values indicate higher association. **Note that these are the logged p-values, natural logarithm of the p-values, and not the p-values.**

```
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 performed 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 subsequent 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.

```
n.tests
```

If you have set hash = TRUE, then the number of tests performed by SES or MMPC will be returned. If you have not set this to TRUE, the number of univariate associations will be returned. So be careful with this number.

```
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.

Generic Functions implemented for 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 <cathineou@csd.uoc.gr> Vincenzo Lagani <vlagani@csd.uoc.gr>

**References**


Fit a mixture of beta distributions in p-values

Description

Fit a mixture of beta distributions in p-values.

Usage

pval.mixbeta(p)

Arguments

p  A vector of p-values.

Details

The p-values are assumed to follow a mixture of two beta distributions. The null p-values follow Be(1, 1) and the non-null p-values follow Be(ξ, 1). In the first step, the proportion of true null values using Storey and Tibshirani (2003) is calculated and then MLE is adopted to obtain ξ. For more information on this see Triantafillou (2014).

Value

A vector with the estimated π₀ and ξ values.

Author(s)

Michail Tsagris

R implementation and documentation: Michail Tsagris <mtsagris@uoc.gr>
Forward Backward Early Dropping selection regression

References


See Also

pc.skel, mmhc.skel, corfs.network, local.mmhc.skel, conf.edge.lower

Examples

```r
## simulate a dataset with continuous data
y <- rdag2(400, p = 25, nei = 3)
ind <- sample(1:25, 25)
x <- y$x[, ind]
mod <- pc.skel(x, method = "comb.fast", alpha = 0.01)
pval <- as.vector(mod$pvalue[lower.tri(mod$pvalue)])
pval.mixbeta(pval)
```

Forward Backward Early Dropping selection regression

Description

Forward Backward Early Dropping selection regression.

Usage

```r
fbed.reg(target, dataset, prior = NULL, ini = NULL, test = NULL, threshold = 0.05, wei = NULL, K = 0, method = "LR", gam = NULL, backward = TRUE)
```

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 = samples).
- `prior`: If you have prior knowledge of some variables that must be in the variable selection phase add them here. This an be a vector (if you have one variable) or a matrix (if you more variables). This does not work during the backward phase at the moment.
If you already have the test statistics and the p-values of the univariate associations (the first step of FBED) supply them as a list with the names "stat" and "pvalue" respectively. If you have used the EBIc this list contains the eBIC of the univariate associations. Note, that the "gam" argument must be the same though.

The available tests: "testIndReg", "testIndPois", "testIndNB", "testIndLogistic", "testIndMMReg", "testIndBinom", "censIndCR", "censIndWR", "testIndBeta", "testIndZIP", "testIndGamma", "testIndNormLog", "testIndTobit", "testIndQPois", "testIndQBinom", "testIndFisher", "testIndMultinom", "testIndOrdinal", "testIndClogit" and "testIndSPML". Note that not all of them work with eBIC. The only test that does not have "eBIC" and no weights (input argument "wei") is the test "gSquare". The tests "testIndFisher" and "testIndSPML" have no weights.

Threshold (suitable values in (0, 1)) for assessing p-values significance. Default value is 0.05.

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. If you want to use the "testIndBinom", then supply the successes in the y and the trials here. An example where weights are used is surveys when stratified sampling has occurred.

How many times should the process be repeated? The default value is 0. You can also specify a range of values of K, say 0:4 for example.

Do you want the likelihood ratio test to be performed ("LR" is the default value) or perform the selection using the "eBIC" criterion (BIC is a special case).

In case the method is chosen to be "eBIC" one can also specify the gamma parameter. The default value is "NULL", so that the value is automatically calculated.

After the Forward Early Dropping phase, the algorithm proceeds with the usual Backward Selection phase. The default value is set to TRUE. It is advised to perform this step as maybe some variables are false positives, they were wrongly selected.

The backward phase using likelihood ratio test and eBIC are two different functions and can be called directly by the user. SO, if you want for example to perform a backard regression with a different threshold value, just use these two functions separately.

The algorithm is a variation of the usual forward selection. At every step, the most significant variable enters the selected variables set. In addition, only the significant variables stay and are further examined. The non significant ones are dropped. This goes until no variable can enter the set. The user has the option to re-do this step 1 or more times (the argument K). In the end, a backward selection is performed to remove falsely selected variables. Note that you may have specified, for example, K=10, but the maximum value FBED used can be 4 for example.

The "testIndQPois" and "testIndQBinom" do not work for method = "eBIC" as there is no BIC associated with quasi Poisson and quasi binomial regression models.

If you specify a range of values of K it returns the results of fbed.reg for this range of values of K. For example, instead of running fbed.reg with K=0, K=1, K=2 and so on, you can run fbed.reg
with K=0:2 and the selected variables found at K=2, K=1 and K=0 are returned. Note also, that
you may specify maximum value of K to be 10, but the maximum value FBED used was 4 (for
example).

Value

If K is a single number a list including:

univ If you have used the log-likelihood ratio test this list contains the test statistics
and the associated logged p-values of the univariate associations tests. If you
have used the EBIC this list contains the eBIC of the univariate associations.
Note, that the "gam" argument must be the same though.

res A matrix with the selected variables, their test statistic and the associated logged
p-value.

info A matrix with the number of variables and the number of tests performed (or
models fitted) at each round (value of K). This refers to the forward phase only.

runtime The runtime required.

back.rem The variables removed in the backward phase.

back.n.tests The number of models fitted in the backward phase.

If K is a sequence of numbers a list tincluding:

univ If you have used the log-likelihood ratio test this list contains the test statistics
and the associated p-values of the univariate associations tests. If you have used
the EBIC this list contains the eBIC of the univariate associations.

res The results of fbed.reg with the maximum value of K.

mod A list where each element refers to a K value. If you run FBEd with method
= "LR" the selected variables, their test statistic and their p-value is returned.
If you run it with method = "eBIC", the selected variables and the eBIC of the
model with those variables are returned.

Author(s)

Michail Tsagris

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

References


7: 1505

See Also

fs.reg, ebic.bsreg, bic.fsreg, MMPC
Examples

```r
# simulate a dataset with continuous data
dataset <- matrix( runif(100 * 20, 1, 100), ncol = 20 )
# define a simulated class variable
target <- rt(100, 10)

a1 <- fbed.reg(target, dataset, test = "testIndReg")
y <- rpois(100, 10)
a2 <- fbed.reg(y, dataset, test = "testIndPois")
a3 <- MMPC(y, dataset)
```

Description

Forward Backward Early Dropping selection regression for big data.

Usage

```r
big.fbed.reg(target = NULL, dataset, threshold = 0.01, ini = NULL, test = "testIndLogistic", K = 0, 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. This can also be NULL and will be extracted from the big.matrix object "dataset". If you want to use the test "censIndWR", for survival data for example, the target must not contain any censored values.

dataset The dataset; this is a big.matrix object, where rows denote the samples and columns the features. If "target" is NULL, the first column must be the target. Only continuous variables are allowed. Note: In the case of the test being "gSquare", the dataset should contain the target variable in the last line.

threshold Threshold (suitable values in (0, 1)) for assessing p-values significance. Default value is 0.05.

ini If you already have the test statistics and the p-values of the univariate associations (the first step of FBED) supply them as a list with the names "stat" and "pvalue" respectively.

test The available tests: "testIndFisher", "testIndPois", "testIndLogistic", "censIndWR", "testIndQPois", "testIndMultinom", "gSquare".

K How many times should the process be repeated? The default value is 0. You can also specify a range of values of K, say 0:4 for example.
Backward Early Dropping selection regression for big data

backward: After the Forward Early Dropping phase, the algorithm proceeds with the usual Backward Selection phase. The default value is set to TRUE. It is advised to perform this step as maybe some variables are false positives, they were wrongly selected. Pay attention to this, as it will convert the big.matrix object with the selected features into a matrix object in R.

The backward phase using likelihood ratio test is a different function and can be called directly by the user. So, if you want to perform a backward regression with a different threshold value, just use that function separately.

Details

The algorithm is a variation of the usual forward selection. At every step, the most significant variable enters the selected variables set. In addition, only the significant variables stay and are further examined. The non-significant ones are dropped. This goes until no variable can enter the set. The user has the option to re-do this step 1 or more times (the argument K). In the end, a backward selection is performed to remove falsely selected variables. Note that you may have specified, for example, K=10, but the maximum value FBED used can be 4 for example.

Notes: The backward phase needs caution, because the big.matrix object with the selected features is turned into a matrix and then the backward selection takes place. In general, this algorithm is to be used with a few tens or hundreds of features and millions of rows. It is designed for thin matrices only. The big.gomp on the other hand is designed for thin, fat and matrices with many rows and many columns.

Value

If K is a single number a list including:

univ: If you have used the log-likelihood ratio test this list contains the test statistics and the associated p-values of the univariate associations tests. If you have used the EBIC this list contains the eBIC of the univariate associations. Note, that the "gam" argument must be the same though.

res: A matrix with the selected variables, their test statistic and the associated logged p-value.

info: A matrix with the number of variables and the number of tests performed (or models fitted) at each round (value of K). This refers to the forward phase only.

runtime: The runtime required.

back.rem: The variables removed in the backward phase.

back.n.tests: The number of models fitted in the backward phase.

Author(s)

Michail Tsagris

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

References

Forward Backward Early Dropping selection regression with GEE

Description

Forward Backward Early Dropping selection regression with GEE.

Usage

fbed.gee.reg(target, dataset, id, prior = NULL, reps = NULL, ini = NULL,
    threshold = 0.05, wei = NULL, K = 0, test = "testIndGEEReg",
corr = "exchangeable", se = "jack")

Arguments

  target  The class variable. This can be a numerical vector with continuous data, binary,
          discrete or an ordered factor variable. It can also be a factor variable with two
          levels only. Pay attention to this. If you have ordinal data, then supply an ordered
          factor variable. The data must be sorted so that observations on a cluster are
          contiguous rows for all entities in the formula.

  dataset The set of candidate predictor variables provide a numerical matrix or a data.frame
          in case of categorical variables (columns = variables, rows = samples). In the
          case of ordinal regression, this can only be a numerical matrix, i.e. only con-
          tinuous predictor variables. The data must be sorted so that observations on a
          cluster are contiguous rows for all entities in the formula.

  id       This is a numerical vector of the same size as target denoting the groups or the
          subjects. The data must be sorted so that observations on a cluster are contiguous
          rows for all entities in the formula.
Forward Backward Early Dropping selection regression with GEE

prior
If you have prior knowledge of some variables that must be in the variable selection phase add them here. This can be a vector (if you have one variable) or a matrix (if you have more variables).

reps
This is a numerical vector of the same size as target denoting the groups or the subjects. If you have longitudinal data and you know the time points, then supply them here. The data must be sorted so that observations on a cluster are contiguous rows for all entities in the formula.

ini
If you already have the test statistics and the logged p-values of the univariate associations (the first step of FBED) supply them as a list with the names "stat" and "pvalue" respectively.

threshold
Threshold (suitable values in (0, 1)) for asmmmbing 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 mentioned in the "geepack" that weights is not (yet) the weight as in sas proc genmod, and hence is not recommended to use.

K
How many times should the process be repeated? The default value is 0.

test
This is for the type of regression to be used, "testIndGEEReg", for Gaussian regression, "testIndGEEGamma" for Gamma regression, "testIndGEELogistic for logistic regression, "testIndGEENormLog" for linear regression with a log link, "testIndGEEPois" for Poisson regression or "testIndGEEOrdinal" for ordinal regression.

correl
The correlation structure. For the Gaussian, Logistic, Poisson and Gamma regression this can be either "exchangeable" (compound symmetry, suitable for clustered data) or "ar1" (AR(1) model, suitable for longitudinal data). If you want to use the AR(1) correlation structure you must be careful with the input dat you give. All data (target and dataset) must be sorted according to the id and time points. All observations must be ordered according to time within each subject. See the vignette of geepack to understand how. For the ordinal logistic regression its only the "exchangeable" correlation structure.

se
The method for estimating standard errors. This is very important and crucial. The available options for Gaussian, Logistic, Poisson and Gamma regression are: a) 'san.se', the usual robust estimate. b) 'jack': if approximate jackknife variance estimate should be computed. c) 'jls': if 1-step jackknife variance estimate should be computed and d) 'fij': logical indicating if fully iterated jackknife variance estimate should be computed. If you have many clusters (sets of repeated measurements) "san.se" is fine as it is asymptotically correct, plus jackknife estimates will take longer. If you have a few clusters, then maybe it's better to use jackknife estimates.

The jackknife variance estimator was suggested by Paik (1988), which is quite suitable for cases when the number of subjects is small (K < 30), as in many biological studies. The simulation studies conducted by Ziegler et al. (2000) and Yan and Fine (2004) showed that the approximate jackknife estimates are in many cases in good agreement with the fully iterated ones.
Details

The algorithm is a variation of the usual forward selection. At every step, the most significant variable enters the selected variables set. In addition, only the significant variables stay and are further examined. The non significant ones are dropped. This goes until no variable can enter the set. The user has the option to redo this step 1 or more times (the argument K). In the end, a backward selection is performed to remove falsely selected variables.

Since GEE are likelihood free, all significance tests take place using the Wald test, hence we decided not to have a backward phase. This algorithm is suitable for both clustered and longitudinal (glmm) data.

If you specify a range of values of K it returns the results of fbed.reg for this range of values of K. For example, instead of running fbed.reg with K=0, K=1, K=2 and so on, you can run fbed.reg with K=0:2 and the selected variables found at K=2, K=1 and K=0 are returned. Note also, that you may specify maximum value of K to be 10, but the maximum value FBED used was 4 (for example).

For GEE make sure you load the library geepack first.

Value

If K is a single number a list including:

- res
  - A matrix with the selected variables, their test statistic and the associated logged p-value.
- info
  - A matrix with the number of variables and the number of tests performed (or models fitted) at each round (value of K).
- runtime
  - The runtime required.

If K is a sequence of numbers a list tincluding:

- univ
  - If you have used the log-likelihood ratio test this list contains the test statistics and the associated logged p-values of the univariate associations tests. If you have used the EBIc this list contains the eBIC of the univariate associations.
- res
  - The results of fbed.gee.reg with the maximum value of K.
- mod
  - A list where each element refers to a K value. If you run FBEd with method = "LR" the selected variables, their test statistic and their p-value is returned. If you run it with method = "eBIC", the selected variables and the eBIC of the model with those variables are returned.

Author(s)

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

References


The R package \texttt{geepack} vignette.

See Also

\texttt{fbed.glmm.reg, glmm.bsreg, MMPC.glmm, fbed.reg}

Examples

```r
## Not run:
require(lme4)
data(sleepstudy)
reaction <- sleepstudy$Reaction
days <- sleepstudy$Days
subject <- sleepstudy$Subject
x <- matrix(rnorm(180 * 200), ncol = 200) ## unrelated predictor variables
m1 <- fbed.gee.reg(reaction, x, subject)
m2 <- fbed.glmm.reg(reaction, x, subject, backward = FALSE)
## End(Not run)
```

Description

Forward Backward Early Dropping selection regression with GLMM.

Usage

\texttt{fbed.glmm.reg(target, dataset, id, prior = NULL, reps = NULL, ini = NULL, threshold = 0.05, wei = NULL, K = 0, method = "LR", gam = NULL, backward = TRUE, test = "testIndGLMMReg")}
Arguments

- **target**: The class variable. This can be a numerical vector with continuous data, binary or discrete valued data. It can also be a factor variable with two levels only.
- **dataset**: The dataset; provide a numerical a matrix (columns = variables, rows = observations).
- **id**: This is a numerical vector of the same size as target denoting the groups or the subjects.
- **prior**: If you have prior knowledge of some variables that must be in the variable selection phase add them here. This can be a vector (if you have one variable) or a matrix (if you more variables). This does not work during the backward phase at the moment.
- **reps**: This is a numerical vector of the same size as target denoting the groups or the subjects. If you have longitudinal data and you know the time points, then supply them here.
- **ini**: If you already have the test statistics and the p-values of the univariate associations (the first step of FBED) supply them as a list with the names "stat" and "pvalue" respectively in the case of method = "LR". In the case of method = "eBIC" this is a list with an element called "bic".
- **threshold**: Threshold (suitable values in (0, 1)) for assuming p-values significance. Default value is 0.05.
- **wei**: A vector of weights to be used for weighted regression. The default value is NULL.
- **K**: How many times should the process be repeated? The default value is 0.
- **method**: Do you want the likelihood ratio test to be performed ("LR" is the default value) or perform the selection using the "eBIC" criterion (BIC is a special case).
- **gam**: In case the method is chosen to be "eBIC" one can also specify the *gamma* parameter. The default value is "NULL", so that the value is automatically calculated.
- **backward**: After the Forward Early Dropping phase, the algorithm proceeds with the usual Backward Selection phase. The default value is set to TRUE. It is advised to perform this step as maybe some variables are false positives, they were wrongly selected.

The backward phase using likelihood ratio test and eBIC are two different functions and can be called directly by the user. So, if you want for example to perform a backward regression with a different threshold value, just use these two functions separately.

- **test**: This is for the type of regression to be used, "testIndGLMMReg", for Gaussian regression, "testIndGLMMLogistic" for logistic regression or "testIndGLMMPois" for Poisson regression, "testIndGLMMGamma" for Gamma regression and "testIndGLMMNormLog" for Gaussian regression with log link and "testIndGLMMOrdinal" for ordinal regression.
Forward Backward Early Dropping selection regression with GLMM

Details

The algorithm is a variation of the usual forward selection. At every step, the most significant variable enters the selected variables set. In addition, only the significant variables stay and are further examined. The non significant ones are dropped. This goes until no variable can enter the set. The user has the option to redo this step 1 or more times (the argument K). In the end, a backward selection is performed to remove falsely selected variables.

Bear in mind that for the "gaussian" case, the forward phase takes place using the F test (Wald statistic calibrated against an F distribution). The backward phase though takes place using the log-likelihood ratio test.

If you specify a range of values of K it returns the results of fbed.reg for this range of values of K. For example, instead of running fbed.reg with K=0, K=1, K=2 and so on, you can run fbed.reg with K=0:2 and the selected variables found at K=2, K=1 and K=0 are returned. Note also, that you may specify maximum value of K to be 10, but the maximum value FBED used was 4 (for example).

Value

If K is a single number a list including:

- `res` A matrix with the selected variables, their test statistic and the associated logged p-value.
- `info` A matrix with the number of variables and the number of tests performed (or models fitted) at each round (value of K). This refers to the forward phase only.
- `runtime` The runtime required.
- `back.rem` The variables removed in the backward phase.
- `back.n.tests` The number of models fitted in the backward phase.

If K is a sequence of numbers a list including:

- `univ` If you have used the log-likelihood ratio test this list contains the test statistics and the associated p-values of the univariate associations tests. If you have used the EBIC this list contains the eBIC of the univariate associations.
- `res` The results of fbed.glmm.reg with the maximum value of K.
- `mod` A list where each element refers to a K value. If you run FBED with method = "LR" the selected variables, their test statistic and their p-value is returned. If you run it with method = "eBIC", the selected variables and the eBIC of the model with those variables are returned.

Author(s)

Michail Tsagris

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


See Also

fbed.gee.reg, glmm.bsreg, MMPC.glmm, cv.fbed.lmm.reg

Examples

```r
## Not run:
require(lme4)
data(sleepstudy)
reaction <- sleepstudy$Reaction
subject <- sleepstudy$Subject
x <- matrix(rnorm(180 * 200), ncol = 200) ## unrelated predictor variables
m1 <- fbed.glmm.reg(reaction, cbind(x1, x), subject)
m2 <- MMPC.glmm(target = reaction, group = subject, dataset = x)

## End(Not run)
```

Description

Variable selection in regression models with forward selection

Usage

```r
fs.reg(target, dataset, ini = NULL, threshold = 0.05, wei = NULL, test = NULL, user_test = 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 available, either all data are continuous, or categorical.
- `ini`: If you have a set of variables already start with this one. Otherwise leave it NULL.
- `threshold`: Threshold (suitable values in (0, 1)) for asssmbsing p-values significance. Default value is 0.05.
wei  A vector of weights to be used for weighted regression. The default value is NULL. An example where weights are used is surveys when stratified sampling has occurred.

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" and "testIndSpearman". If you have continuous predictor variables in matrix form, you can put "testIndFisher" and this is pretty fast. Instead of calculating partial F-tests, which requires linear regression models to be fit, it calculates partial correlation coefficients and this is much more efficient.

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 squared is used.

tol  The difference between two successive values of the stopping rule. By default this is set to 2. If for example, the BIC difference between two successive models is less than 2, the process stops and the last variable, even though significant does not enter the model.

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 an 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 and the dataset is not a matrix, but a data.frame linear regression is used. If however, the dataset is a matrix, the correlation based forward selection is used. That is, instead of partial F-tests, we do partial correlation tests.
- 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

In the case of test="testIndMMReg" and class(dataset) = matrix, just one matrix is returned with the index of the selected variable(s), the p-value, the test statistic and the BIC value of each model. For all other cases, 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 **logged p-values**.

**info**  
A matrix with the selected variables, their **logged 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.

**ci_test**  
The conditional independence test used.

**final**  
The final regression model.

**Author(s)**

Michail Tsagris

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

**See Also**

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

**Examples**

```r
set.seed(123)

# simulate a dataset with continuous data
dataset <- matrix( runif(500 * 20, 1, 100), ncol = 20 )

define a simulated class variable
target <- rt(500, 10)

a0 <- fs.reg(target, dataset, threshold = 0.05, stopping = "BIC", tol = 2)

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

require(survival, quietly = TRUE)
y <- survival::Surv(rexp(500), rep(1, 500) )
a2 <- fs.reg(y, dataset, threshold = 0.05, test = "censIndWR", stopping = "BIC",
tol = 2)
a3 <- MMPC(target, dataset)

target <- factor( rbinom(500, 1, 0.6) )
b2 <- fs.reg(target, dataset, threshold = 0.05, test = NULL, stopping = "BIC", tol = 2)
```
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

```r
glm.fsreg(target, dataset, ini = NULL, threshold = 0.05, wei = NULL, tol = 2, ncores = 1)
gamma.fsreg(target, dataset, ini = NULL, threshold = 0.05, wei = NULL, tol = 2, ncores = 1)
normlog.fsreg(target, dataset, ini = NULL, threshold = 0.05, wei = NULL, tol = 2, ncores = 1)
```

Arguments

target  
A numerical vector or a factor variable with two levels. The Gamma regression requires strictly positive numbers, whereas the normlog requires positive numbers, zero included.
dataset  
The dataset; provide either a data frame or a matrix (columns = variables, rows = samples). In either case, only two cases are available, either all data are continuous, or categorical.
ini  
If you have a set of variables already start with this one. 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. An example where weights are used is surveys when stratified sampling has occurred.
tol  
The difference between two successive values of the stopping rule. By default this is set to 2. If for example, the BIC difference between two successive models is less than 2, the process stops and the last variable, even though significant does not enter the model.
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 cammb 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 **logged p-values**. If you have logistic or Poisson regression with continuous predictor variables in a matrix form and no weights, this will not appear. In this case, a C++ code is called and the output is less.

- `info`: A matrix with the selected variables, their **logged 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.

- `ci_test`: The conditional independence test used.

- `final`: The final regression model.

**Author(s)**

Michail Tsagris

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

**See Also**

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

**Examples**

```r
set.seed(123)

# simulate a dataset with continuous data
dataset <- matrix( runif(200 * 30, 1, 100), ncol = 30 )

define a simulated class variable
target <- rpois(200, 10)

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

**Description**

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

Usage

```r
lm.fsreg(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 available, 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. An example where weights are used is surveys when stratified sampling has occurred.
- `stopping` The stopping rule. The BIC ("BIC") or the adjusted $R^2$ ("adjrsq") can be used.
- `tol` The difference between two successive values of the stopping rule. By default this is set to 2. If for example, the BIC difference between two successive 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.
- `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 commands 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.

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 logged p-values.
**Description**

The main task of this test is to provide a p-value \( PV \) 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**

```r
G-square conditional independence test for discrete data

```
G-square conditional independence test for discrete data

Arguments

**target**  
A numeric vector containing the values of the target variable. **The minimum value must be 0.**

**dataset**  
A numeric matrix containing the variables for performing the test. Rows as samples and columns as features. **The minimum value must be 0.**

**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**  
This argument is not used in this test.

**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 successful) 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 which contains the cached generated statistics of a SES run in the current dataset, using the current test.

**pvalue_hash**  
A hash object which contains the cached generated p-values of a SES run in the current dataset, using the current test.

**threshold**  
Threshold (suitable values in (0, 1)) for assessing p-values significance. Default value is 0.05. This is actually obsolete here, but has to be in order to have a concise list of input arguments across the same family of functions.

**R**  
The number of permutations to use. The default value is 999.

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).
Generalised linear mixed models based on glmm SES and MMPC outputs

stat          A numeric value that represents the generated statistic of the $G^2$ test (see reference below).
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

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");

Generalised linear mixed models based on glmm SES and MMPC outputs

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

```r
mmpc.glmm.model(target, dataset, reps = NULL, group, slopes = FALSE, wei = NULL,
                  mmpcglmm.Object, test = NULL)

ses.glmm.model(target, dataset, reps = NULL, group, slopes = FALSE, wei = NULL,
                 sesglmm.Object, nsignat = 1, test = NULL)

mmpc.gee.model(target, dataset, reps = NULL, group, correl = "exchangeable",
                 se = "jack", wei = NULL, mmpcgee.Object, test = NULL)

ses.gee.model(target, dataset, reps = NULL, group, correl = "exchangeable",
              se = "jack", wei = NULL, sesgee.Object, nsignat = 1, test = NULL)
```

Arguments

target  The class variable. Provide a vector with continuous (normal), binary (binomial) or discrete (Poisson) data.
dataset The dataset; provide either a data frame or a matrix (columns = variables, rows = observations). Currently, only continuous datasets are supported.
reps A numeric vector containing the time points of the subjects. Its 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 length as target.
slopes Should random slopes for the time effect be fitted as well? Default value is FALSE.
correl The correlation structure. For the Gaussian, Logistic, Poisson and Gamma regression this can be either "exchangeable" (compound symmetry, suitable for clustered data) or "ar1" (AR(1) model, suitable for longitudinal data). For the ordinal logistic regression its only the "exchangeable" correlation structure.
se The method for estimating standard errors. This is very important and crucial. The available options for Gaussian, Logistic, Poisson and Gamma regression are: a) 'san.se', the usual robust estimate. b) 'jack': if approximate jackknife variance estimate should be computed. c) 'jls': if 1-step jackknife variance estimate should be computed and d) 'fij': logical indicating if fully iterated jackknife variance estimate should be computed. If you have many clusters (sets of repeated measurements) "san.se" is fine as it is asymptotically correct, plus jackknife estimates will take longer. If you have a few clusters, then maybe it's better to use jackknife estimates.
wei A vector of weights to be used for weighted regression. The default value is NULL.
Generalised linear mixed models based on glmm SES and MMPC outputs

mmpcglmm.Object
An object with the results of an MMPC.glmm run.

sesglmm.Object
An object with the results of a SES.glmm run.

mmpcgee.Object
An object with the results of an MMPC.gee run.

sesgee.Object
An object with the results of an SES.gee 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 number 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
The conditional independence test to use. Default value is NULL. Currently, the only available conditional independence test are the testIndGLMMReg, testIndGLMMLogistic, testIndGLMMPois and testIndLMM which fit linear mixed models.

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

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: Michail Tsagris <mtsagris@uoc.gr>

References

See Also
SES, MMPC, cv.ses, cv.mmpc
Examples

```r
## Not run:
require(lme4)
data(sleepstudy)
reaction <- sleepstudy$Reaction
days <- sleepstudy$Days
subject <- sleepstudy$Subject
x <- matrix(rnorm(180 * 50), ncol = 50)  ## unrelated predictor variables
m1 <- SES.glmm(reaction, days, subject, x)
m2 <- MMPC.glmm(reaction, days, subject, x)
mod <- mmpc.glmm.model(reaction, dataset = x, reps = days, group = subject, slopes = FALSE, mmpcglmm.Object = m2)
## End(Not run)
```

Generalised ordinal regression

Generalised ordinal regression.

Usage

`ordinal.reg(formula, data)`

Arguments

- `formula`: An object of class "formula" (or one that can be coerced to that class): a symbolic description of the model to be fitted. This is the usual formula use by many regression models in R and other packages. Type "glm" or "formula" in R for more information.
- `data`: A data.frame object carrying the relevant data.

Details

Generalised ordinal regression is fitted. This means the instead of having the same coefficient for each predictor variable, they are allowed to vary. The usual, proportional odds, ordinal regression specifies that the lines do not cross. This one does not need the proportional odds assumption. The proportional odds assumption does not always hold in practice and is a rather restrictive model. Be careful though, you may end up with negative probabilities. We do a tick to fix them, but in that case, you may have not found the optimal model. This is a problematic case unfortunately. Williams (2006) explains in a very nice way how one can fit this model by using many logistic regressions in an incremental way. The number of logistic regression models to be fit is the number of categories of the response variables - 1.

It may be the case that the message says "problematic region". In this case the optimization was not succesful and perhaps the deviance is not at the global minimum. For example, with the addition of
one extra variable the deviance might increase. I know, this type of ordinal regression is hard. In these difficult situations other packages return "error". Another difficult I have seen is when "NA" appear in the coefficients. In this case I do not consider these coefficients, not their corresponding variables in the calculation of the deviance.

Value

A list including:

- message: If you hit negative probabilities, the message "problematic region" will be printed. Otherwise, this is NULL.
- be: The regression coefficients.
- devi: The deviance of the model.

Author(s)

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

References


See Also

pc.skel, ridge.reg

Examples

```r
y <- factor(rbinom(100, 3, 0.5))
x <- matrix(rnorm(100 * 3), ncol = 3)
ordinal.reg(y ~ x, data = data.frame(x))
ordinal.reg(y ~ 1, data = data.frame(x))
```

Usage

generatefolds(target, nfolds = 10, stratified = TRUE, seed = FALSE)
Generic orthogonal matching pursuit (gOMP)

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>target</td>
<td>A vector with some data, either continuous or categorical.</td>
</tr>
<tr>
<td>nfolds</td>
<td>The number of folds to produce.</td>
</tr>
<tr>
<td>stratified</td>
<td>A boolean variable specifying whether stratified random (TRUE) or simple random (FALSE) sampling is to be used when producing the folds.</td>
</tr>
<tr>
<td>seed</td>
<td>A boolean variable. If set to TRUE, the folds will always be the same.</td>
</tr>
</tbody>
</table>

**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 element is a fold containing the indices of the data.

**Author(s)**

Michail Tsagris

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

**See Also**

`cv.ses`

**Examples**

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

---

**Description**

Generic orthogonal matching pursuit.

**Usage**

```r
gomp(target, dataset, xstand = TRUE, tol = qchisq(0.95, 1),
       test = "testIndLogistic", method = "ar2")

gomp.path(target, dataset, xstand = TRUE, tol = c(4, 5, 6),
           test = "testIndLogistic", method = "ar2")

boot.gomp(target, dataset, tol = qchisq(0.95, 1),
          test = "testIndLogistic", method = "ar2", B = 500, ncores = 1)
```
Generic orthogonal matching pursuit (gOMP)

Arguments

- **target**: The response variable, a numeric vector, a matrix or a Surv object.
- **dataset**: A matrix with continuous data, where the rows denote the samples and the columns are the variables.
- **xstand**: If this is TRUE the independent variables are standardised.
- **tol**: The tolerance value to terminate the algorithm. This is the change in the criterion value between two successive steps. The default value is the 95% quantile of the $\chi^2$ distribution with 1 degree of freedom. For `test = "testIndFisher"` the BIC is already calculated.

In the case of "gomp.path" this is a vector of values. For each tolerance value the result of the gOMP is returned. It returns the whole path of solutions.

- **test**: This denotes the parametric model to be used each time. It depends upon the nature of the target variable. The possible values are "testIndFisher" (or "testIndDReg" for the same purpose), "testIndLogistic", "testIndPois", "testIndQPoiss", "testIndQbinom", "testIndNormLog", "testIndMVreg", "testIndNB", "testIndBeta", "testIndGamma", "testIndMMReg", "testIndRQ", "testIndOrdinal", "testIndTobit", "censIndCR", "censIndWR", "censIndLLR" and "testIndMultinom".

- **method**: This is only for the "testIndFisher". You can either specify, "ar2" for the adjusted R-square or "sse" for the sum of squares of errors. The tolerance value in both cases must a number between 0 and 1. That will denote a percentage. If the percentage increase or decrease is less than the number the algorithm stops. An alternative is "BIC" for BIC and the tolerance values are like in all other regression models.

- **B**: How many bootstrap samples to generate. The gOMP will be performed for each of these samples.

- **ncores**: How many cores to use. This argument is valid only if you have a multi-threaded machine.

Value

A list including:

- **runtime**: The runtime of the algorithm
- **phi**: The $\phi$ coefficient, returned in the quasi binomial (testIndQBinom), quasi Poisson (testIndQPoiss), Gamma (testIndGamma) and Gaussian with log link (testIndNormLog). In all other cases this is NULL.
- **res**: For the case of "gomp" a matrix with two columns. The selected variable(s) and the criterion value at every step. For the case of "gomp.path" a matrix with many columns. Every column contains the selected variables for each tolerance value, starting from the smallest value (which selected most variables). The final column is the deviance of the model at each step. For the "boot.gomp" this is a matrix with two columns. The first one is the selected variables and the second column is their proportion of selection.
Author(s)

Michail Tsagris

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

References


See Also

cor.fbed, cor.fsreg, correls, fs.reg

Examples

```r
x <- matrix( rnorm(500 * 50), ncol = 50)
y <- rnorm(500)
b <- MXM::gomp(y, x, test = "testIndFisher")
```
Generic orthogonal matching pursuit (gOMP) for big data

Description

Generic orthogonal matching pursuit (gOMP) for big data.

Usage

big.gomp(target = NULL, dataset, tol = qchisq(0.95, 1) + log(dim(x)[1]),
          test = "testIndFisher", method = "ar2")

big.gomp.path(target = NULL, dataset, tol = c(8, 9, 10),
               test = "testIndFisher", method = "ar2")

Arguments

target

The target (response) variable. If NULL, then it must inside the dataset. You
might have the target variable though outside the big data file. This is like in
the case of the regular gomp, a Surv object, a factor or a continuous numerical
vector for all other cases. The default value is NULL.

dataset

The big.matrix oject. If target is NULL, the first column must be the target, the
response variable and all the others are the predictor variables. In the case of
survival data, the first two columns are used to form the response variable.

tol

The tolerance value to terminate the algorithm. This is the change in the criterion
value between two successive steps. The default value is the 95\% quantile of the
\( \chi^2 \) distribution with 1 degree of freedom. For test = "testIndFisher" the BIC is
already calculated.

In the case of "big.gomp.path" this is a vector of values. For each tolerance
value the result of the gOMP is returned. It returns the whole path of solutions.

test

This denotes the parametric model to be used each time. It depends upon the
nature of the target variable. The possible values are "testIndFisher" (or "testInd-
logistic", "testIndPois", "testIndQPois", "testIndQbinom", "testIndNormLog", "testIndNB", "testInd-
Gamma", "testIndMMReg", "testIndRQ", "testIndOrdinal", "testIndTobit", "censIndCR" and "censIndWR".

method

This is only for the "testIndFisher". You can either specify, "ar2" for the adjusted
R-square or "sse" for the sum of squares of errors. The tolerance value in both
cases must a number between 0 and 1. That will denote a percentage. If the
percentage increase or decrease is less than the nubmer the algorithm stops.
An alternative is "BIC" for BIC and the tolerance values are like in all other
regression models.
Generic orthogonal matching pursuit (gOMP) for big data

Details

The data (matrix) which will be read and compressed into a big.matrix object must be of type "numeric". We tested it and it works with "integer" as well. But, in general, bear in mind that only matrices will be read. We have not tested with data.frame for example. Whatasoever, in the help page of the package "bigmemory" this is mentioned: Any non-numeric entry will be ignored and replaced with NA, so reading something that traditionally would be a data.frame won’t cause an error. A warning is issued. In all cases, the object size is always 696 bytes!

Value

A list including:

- runtime: The runtime of the algorithm
- phi: The $\phi$ coefficient, returned in the quasi binomial (testIndQBinom), quasi Poisson (testIndQPois), Gamma (testIndGamma) and Gaussian with log link (testIndNormLog). In all other cases this is NULL.
- res: For the case of "big.gomp" a matrix with two columns. The selected variable(s) and the criterion value at every step. For the case of "gomp.path" a matrix with many columns. Every column contains the selected variables for each tolerance value, starting from the smallest value (which selected most variables). The final column is the deviance of the model at each step.

Author(s)

Michail Tsagris

R implementation and documentation: Michail Tsagris <mtsagris@uoc.gr>. For more information see the "bigmemory" package.

References


See Also

gomp, read.big.data

Examples

```r
## Not run:
dataset <- matrix( runif(10^6 * 50, 1, 100), ncol = 50 )
write.csv(data, "dataset.csv", sep = ",")
a <- read.big.data("dataset.csv")
mod <- big.gomp(dataset = a, test = "testIndFisher", tol = 0.01)
## End(Not run)
```

Graph of unconditional associations

```r
Graph of unconditional associations
```

Description

Calculates the graph of unconditional associations. If the correlation (Pearson, Spearman) or the \( G^2 \) test of independence, between pairs of continuous or categorical variables respectively is not statistically significant, there is no edge between the two respective nodes.

Usage

```r
corgraph(dataset, test = "testIndFisher", threshold = 0.01)
```

Arguments

- `dataset`: A matrix with the variables. The user must know if they are continuous or if they are categorical. If you have a matrix with categorical data, i.e. 0, 1, 2, 3 where each number indicates a category, the minimum number for each variable must be 0.

- `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.

- `threshold`: Threshold (suitable values in (0, 1)) for assessing p-values significance. Default value is 0.05.
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.
- **stat**: A matrix with the test statistics.
- **pvalue**: A matrix with the p-values.
- **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.

Author(s)

Michail Tsagris

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

See Also

[pc.skel](#), [mmhc.skel](#), [corfs.network](#), [bn.skel.utils](#)

Examples

```r
# simulate a dataset with continuous data
y <- rdag2(500, p = 20, nei = 3)
x <- y$x
a <- mmhc.skel(x, max_k = 5, threshold = 0.01, test = "testIndFisher")
b <- pc.skel(x, alpha = 0.01)
d <- corgraph(x, test = "testIndFisher", threshold = 0.01)
```

IAMB backward selection phase

Description

IAMB backward selection phase.

Usage

```r
iamb.bs(target, dataset, threshold = 0.05, wei = NULL, test = NULL, user_test = NULL)
```
IAMB backward selection phase

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 available, 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. An example where weights are used is surveys when stratified sampling has occurred.

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.

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 every 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 logged 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@uoc.gr>

References

IAMB variable selection

See Also

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

Examples

```r
set.seed(123)
dataset <- matrix( runif(500 * 10, 1, 100), ncol = 10 )
target <- rnorm(500)
a1 <- iamb.bs(target, dataset, threshold = 0.05, test = "testIndRQ")
a2 <- bs.reg(target, dataset, threshold = 0.05, test = "testIndRQ")
```

Description

IAMB variable selection.

Usage

```r
iamb(target, dataset, threshold = 0.05, wei = NULL, test = NULL, user_test = NULL,
stopping = "BIC", tol = 2, 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 available, 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. An example where weights are used is when stratified sampling has occurred.

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" and "testIndIGreg".

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.
**IAMB variable selection**

- **tol**: The difference between two successive values of the stopping rule. By default this is set to 2. If for example, the BIC difference between two successive models is less than 2, the process stops and the last variable, even though significant does not enter the model.

- **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 significant variables, the one with the maximum p-value is dropped. So, one variable is removed at every step. In the IAMB backward phase, at every 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 selected 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@uoc.gr>

**References**


**See Also**

`glm.fsreg, lm.fsreg, bic.fsreg, bic glm.fsreg, CondIndTests, MMPC, SES`
Incremental BIC values and final regression model of the FBED algorithm

Examples

```r
set.seed(123)
dataset <- matrix( runif(100 * 50, 1, 100), ncol = 50 )

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")
```

Description

Incremental BIC values and final regression model of the FBED algorithm.

Usage

```r
fbedreg.bic(target, dataset, wei = NULL, fbedreg.object, test = NULL, graph = TRUE)
```

Arguments

- `target`: A numeric vector containing the values of the target variable. 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 or data.frame 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. An example where weights are used is surveys when stratified sampling has occured. We suggest not to use weights if you choose "testIndMMReg" as weights are already being used there.
- `fbedreg.object`: An object with the results of an FBED run.
- `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.
- `graph`: If you want a graphical representation of the drop in the BIC values set this to TRUE.

Details

This function takes the output of the FBED (`fbed.reg`) and fits succesive models calculating the BIC for each of them. A graph can also be returned.
Interactive plot of an (un)directed graph

Value

- **res**: A matrix with the selected variables, their test statistic and p-value (taken from the fbedreg.object) along with the BIC. Each row contains a BIC, that is the BIC of the model with the variables up to that row.
- **mod**: The final model with all selected variables.

Author(s)

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

References


See Also

- `reg.fit`, `mmpc.model`, `SES`, `MMPC`, `cv.ses`, `cv.mmpc`

Examples

```r
dataset <- matrix( runif(100 * 20, 1, 100), ncol = 20 )
# define a simulated class variable
target <- rt(100, 10)
a <- fbed.reg(target, dataset, K = 10, test = "testIndFisher", method = "eBIC")
fbedreg.bic(target, dataset, fbedreg.object = a, test = "testIndFisher")
```

Interactive plot of an (un)directed graph

Description

Interactive plot of an (un)directed graph.

Usage

```r
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".
Lower limit of the confidence of an edge

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, stretch it etc.

Author(s)

Michail Tsagris

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

See Also

mmhc.skel, nei, pc.skel, mb

Examples

## Not run:
# simulate a dataset with continuous data
dataset <- matrix( runif(200 * 20, 1, 100), nrow = 200 )
a <- mmhc.skel(dataset, max_k = 3, threshold = 0.05, test = "testIndFisher",
ne = 1)
plotnetwork(a$G)
plotnetwork(a$G, titlos = "DAG skeleton")
## End(Not run)

Lower limit of the confidence of an edge

Description

Lower limit of the confidence of an edge.

Usage

conf.edge.lower(p)

Arguments

p A numerical vector with the proportion of times an edge was found in the bootstrapped PC algorithm or the confidence of the edge returned by \texttt{bn.skel.utils2}. 

Details

After having performed PC algorithm many times in the bootstrap samples (using `pc.skel.boot` for example) you get a symmetric matrix with the proportion of times an edge was discovered. Take the lower (or upper) triangular elements of that matrix and pass them as input in this function. This will tell you the minimum proportion required to be confident that an edge is truly significant.

Value

The estimated cutoff limit above which an edge can be deemed significant.

Author(s)

Michail Tsagris

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

References


See Also

`pc.skel.boot`

Examples

```r
y <- rdag2(200, p = 40, nei = 3)
x <- y$x
g <- pc.skel.boot(x, R = 199)$Gboot
a <- g[ lower.tri(g) ]
conf.edge.lower(a)
```

Description

`mammpc.output` 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"

Methods

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 approximate simple logistic regressions

Many approximate simple logistic regressions.
Arguments

- **target**: The dependent variable, a numerical vector with 0s or 1s.
- **dataset**: A matrix with the independent variables.
- **logged**: Should the p-values be returned (FALSE) or their logarithm (TRUE)?

Details

Many simple approximate logistic regressions are performed and hypothesis testing for the significance of each coefficient is returned. The code is available in the paper by Sikorska et al. (2013). We simply took the code and made some minor modifications. The explanation and the motivation can be found in their paper. They call it semi-parallel logistic regressions, hence we named the function `sp.logiregs`.

Value

A two-column matrix with the test statistics (Wald statistic) and their associated p-values (or their logarithm).

Author(s)

- Initial author Karolina Sikorska in the above reference paper. Modifications by Michail Tsagris.

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

References


See Also

- `logiquant.regs`, `bic.regs`

Examples

```r
y <- rbinom(200, 1, 0.5)
x <- matrix( rnorm(200 * 30), ncol = 30 )
a <- MXM::sp.logiregs(y, x)
```
Many simple beta regressions

Many simple beta regressions.

Description

Many simple beta regressions.

Usage

\[
\text{beta.regs(target, dataset, wei = NULL, check = FALSE, logged = FALSE, ncores = 1)}
\]

\[
\text{perm.betaregs(target, dataset, wei = NULL, check = FALSE, logged = FALSE, threshold = 0.05, R = 999, ncores = 1)}
\]

\[
\text{wald.betaregs(target, dataset, wei = NULL, check = FALSE, logged = FALSE, ncores = 1)}
\]

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 data frame with continuous only variables, a dataframe with mixed or only categorical variables.
- **wei**: A vector of weights to be used for weighted regression. The default value is NULL. An example where weights are used is surveys when stratified sampling has occured.
- **check**: A boolean variable indicating whether to check for variables with identical values. The default is FALSE.
- **logged**: A boolean variable; it will return the logarithm of the p-value if set to TRUE.
- **threshold**: Threshold (suitable values in (0, 1)) for assessing p-values significance. Default value is 0.05.
- **R**: The number of permutations, set to 999 by default. There is a trick to avoid doing all permutations. As soon as the number of times the permuted test statistic is more than the observed test statistic is more than 50 (in this example case), the p-value has exceeded the significance level (threshold value) and hence the predictor variable is not significant. There is no need to continue do the extra permutations, as a decision has already been made.
- **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.
Many simple quantile regressions using logistic regressions

Author(s)

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

References


See Also

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

Examples

```r
y <- rbeta(80, 5, 3)
x <- matrix( rnorm(80 * 7), ncol = 7)
a1 <- beta.regs(y, x)
a2 <- perm.betaregs(y, x[, 1:4], R = 299)
```

Description

Many simple quantile regressions using logistic regressions.

Usage

```r
logiquant.regs(target, dataset, logged = FALSE)
```

Arguments

- `target` The dependent variable, a numerical vector.
- `dataset` A matrix with the independent variables.
- `logged` Should the p-values be returned (FALSE) or their logarithm (TRUE)?

Details

Instead of fitting quantile regression models, one for each predictor variable and trying to assess its significance, Redden et al. (2004) proposed a simple singificance test based on logistic regression. Create an indicator variable I where 1 indicates a response value above its median and 0 elsewhere. Since I is binary, perform logistic regression for the predictor and assess its significance using the likelihood ratio test. We perform many logistic regression models since we have many predictors whose univariate association with the response variable we want to test.
Many simple zero inflated Poisson regressions

Value
A two-column matrix with the test statistics (likelihood ratio test statistic) and their associated p-values (or their logarithm).

Author(s)
Author: Michail Tsagris.

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

References

See Also

bic.regs, sp.logiregs

Examples
y <- rcauchy(100, 3, 2)
x <- matrix( rnorm(100 * 50), ncol = 50 )
a <- MXM::logiquant.regs(y, x)
Arguments

- **target**: The target (dependent) variable. It must be a numerical vector with integers.
- **dataset**: The independent 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. An example where weights are used is surveys when stratified sampling has occurred.
- **check**: A boolean variable indicating whether to check for variables with identical values. The default is FALSE.
- **logged**: A boolean variable; it will return the logarithm of the pvalue if set to TRUE.
- **R**: The number of permutations, set to 999 by default. There is a trick to avoid doing all permutations. As soon as the number of times the permuted test statistic is more than the observed test statistic is more than 50 (in this example case), the p-value has exceeded the significance level (threshold value) and hence the predictor variable is not significant. There is no need to continue do the extra permutations, as a decision has already been made.
- **threshold**: Threshold (suitable values in (0, 1)) for assessing p-values significance. Default value is 0.05.
- **ncores**: The number of cores to use. The default value is 1.

Details

Many simple zero inflated Poisson regressions are fitted. The permutations based approach may cause some errors sometimes, and this is due to the nature of the distribution and its maximisation process. "nlm" and "optim" are used internally.

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@uoc.gr>

References


See Also

- `zip.mod`, `testIndZIP`, `reg.fit`, `ridge.reg`
Many Wald based tests for logistic and Poisson regressions with continuous predictors

Examples

```r
y <- rpois(50, 3)
x <- matrix( rnorm(50 * 7), ncol = 7)
y[1:10] <- 0
a1 <- zip.regs(y, x)
a2 <- perm.zipregs(y, x[, 1:3], R = 299)
```

Description

Many Wald based tests for logistic and Poisson regressions with continuous predictors.

Usage

```r
wald.logisticregs(y, x, tol = 1e-09, wei = NULL, check = FALSE, logged = FALSE, ncores = 1)
wald.poissonregs(y, x, tol = 1e-09, wei = NULL, check = FALSE, logged = FALSE, ncores = 1)
```

Arguments

- `y`: A vector with either 0s and 1 (logistic regression) or discrete data, counts (Poisson regression).
- `x`: A data.frame, the predictor variables. If you have no categorical variables, the function will still work but it’s better to use the `score.glms` because it is faster.
- `tol`: The tolerance value to stop the Newton-Raphson iterations. It is set to 1e-09 by default.
- `wei`: A vector of weights to be used for weighted regression. The default value is NULL. An example where weights are used is surveys when stratified sampling has occurred.
- `check`: A boolean variable indicating whether to check for variables with identical values. The default is FALSE.
- `logged`: A boolean variable; it will return the logarithm of the pvalue if set to TRUE.
- `ncores`: How many to cores to use; the default value is 1.

Details

Instead of using R built-in function `glm` we implemented the newton-Raphson in order to avoid unnecessary calculations. The functions are much faster.
Markov Blanket of a node in a directed graph

Value
A matrix with three columns, the test statistic, its associated (logged) p-value and the BIC of each model.

Author(s)
Michail Tsagris
R implementation and documentation: Michail Tsagris <mtsagris@uoc.gr> and Manos Papadakis <papadakm95@gmail.com>.

References

See Also
univregs, perm.univregs

Examples
## 20 variables, hence 20 univariate regressions are to be fitted
x <- matrix( rnorm(200 * 20), ncol = 20 )
y <- rpois(200, 4)
a <- wald.poissonregs(y, x)
b <- univregs(y, x, test = testIndPois)
cor(exp(a[, 2]), exp(b$pvalue) )

Markov Blanket of a node in a directed graph

    Returns the Markov blanket of a node (or variable)

Description
Returns the Markov blanket of a node (or variable).

Usage
\texttt{mb(G, node)}

Arguments
\begin{itemize}
\item \texttt{G} \hspace{1cm} The graph matrix as produced from \texttt{pc.or} or any other algorithm which produces directed graphs.
\item \texttt{node} \hspace{1cm} A vector with one or more numbers indicating the seleted node(s) (or variable(s)).
\end{itemize}
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@uoc.gr>

See Also

- plotnetwork, nei, pc.or

Examples

```r
# 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)
```

---

**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"

Methods

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")

Description

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

Usage

```r
mmpc.path(target, dataset, wei = NULL, max_ks = NULL, alphas = NULL, test = NULL, 
user_test = NULL, ncores = 1)
wald.mmpc.path(target, dataset, wei = NULL, max_ks = NULL, alphas = NULL, test = NULL, 
user_test = NULL, ncores = 1)
perm.mmpc.path(target, dataset, wei = NULL, max_ks = NULL, alphas = NULL, test = NULL, 
user_test = NULL, R = 999, 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). 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. An example where weights are used is surveys when stratified sampling 
has occurred.
- `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 = (4,3,2) are used 
by default.
- `alphas` 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.
- `R` The number of permutations, set to 999 by default. There is a trick to avoid do-
ing all permutations. As soon as the number of times the permuted test statistic 
is more than the observed test statistic is more than 50 (if threshold = 0.05 and 
R = 999), the p-value has exceeded the significance level (threshold value) and 
hence the predictor variable is not significant. There is no need to continue do 
the extra permutations, as a decision has already been made.
- `ncores` How many cores to use. This plays an important role if you have tens of thou-
sands 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 algorithm 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 length 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**


**See Also**

CondIndTests, cv.ses

**Examples**

```r
set.seed(123)
# simulate a dataset with continuous data
dataset <- matrix(runif(500 * 51, 1, 100), nrow = 500 )
# the target feature is the last column of the dataset as a vector
target <- dataset[, 51]
dataset <- dataset[, -51]

a <- mmpc.path(target, dataset, max_ks = NULL, alphas = NULL, test = NULL, user_test = NULL, ncores = 1)
```
Class "MMPC.gee.output"

Description

MMPC.glmm output object class.

Objects from the Class

Objects can be created by calls of the form

```r
new("MMPC.gee.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"
- `n.tests`: Object of class "numeric"
- `runtime`: Object of class "proc_time"
- `test`: Object of class "character"
- `correl`: Object of class "character"
- `se`: Object of class "character"

Methods

- `plot` plot(x = "MMPC.gee.output", mode = "all"): Generic function for plotting the generated p-values of the MMPC.glmm.output object. Argument mode = "all" for plotting all the p-values or mode= "partial" for partial plotting the first 500 p-values

Author(s)

Giorgos Athineou and Michail Tsagris <mtsagris@uoc.gr>

See Also

`MMPC.gee`, `SES.gee`

Examples

```r
showClass("MMPC.gee.output")
```
**MMPC.glmm.output-class**

Class "MMPC.glmm.output"

**Description**

MMPC.glmm output object class.

**Objects from the Class**

Objects can be created by calls of the form `new("MMPC.glmm.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"
- n.tests: Object of class "numeric"
- runtime: Object of class "proc_time"
- test: Object of class "character"
- slope: Object of class "logical"

**Methods**

- `plot(x = "MMPC.glmm.output", mode = "all")`: Generic function for plotting the generated pvalues of the MMPC.glmm.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.glmm`, `SES.glmm`

**Examples**

`showClass("MMPC.glmm.output")`
Description

MMPC output object class.

Objects from the Class

Objects can be created by calls of the form \texttt{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"
- n.tests: Object of class "numeric"
- runtime: Object of class "proc_time"
- test: Object of class "character"

Methods

- \texttt{plot} \texttt{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

\texttt{showClass("MMPCoutput")}
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( ... )
- condi.perm( ... )
- InternalSES.glmm( ... )
- InternalMMPC.glmm( ... )
- IdentifyEquivalence.glmm( ... )
- apply_ideq.glmm( ... )
- identifyTheEquivalent.glmm( ... )
- max_min_assoc.glmm( ... )
- min_assoc.glmm( ... )
- univariateScore.glmm( ... )
• is.sepset( ... )
• lm.fsreg_2( ... )
• glm.fsreg_2( ... )
• dag_to_eg( ... )
• nchoosek( ... )
• R0( ... )
• R1( ... )
• R2( ... )
• R3( ... )
• is.sepset( ... )
• regbeta( ... )
• regbetawei( ... )
• betaml.wei( ... )
• regzip( ... )
• regzipawei( ... )
• zipml.wei( ... )
• zipwei( ... )
• bic.betafsreg( ... )
• bic.zipfsreg( ... )
• beta.fsreg( ... )
• zip.fsreg( ... )
• beta.bsreg( ... )
• zip.bsreg( ... )
• iamb.betabs( ... )
• iamb.zipbs( ... )
• iamb.glmbs( ... )
• internaliamb.binombs( ... )
• internaliamb.poisbs( ... )
• internaliamb.lmbs( ... )
• InternalMMPC.glmm( ... )
• InternalSES.glmm( ... )
• InternalSES( ... )
• InternalMMPC( ... )
• univariateScore( ... )
• perm.univariateScore( ... )
• max_min_assoc( ... )
• min_assoc( ... )
• nchoosek( ... )
• Internalmases( ... )
• compare_p_values! ( ... )
• perm.Internalsmpc
• wald.Internalsmpc
• wald.Internalsases
• perm.IdentifyEquivalence( ... )
• perm.identifyTheEquivalent( ... )
• perm.apply_ideq( ... )
• IdentifyEquivalence( ... )
• apply_ideq( ... )
• identifyTheEquivalent( ... )
• cvs.par( ... )
• cvmmmpc.par( ... )
• cvwaldses.par( ... )
• cvwaldmmpc.par( ... )
• cvpermses.par( ... )
• cvpermmmpc.par( ... )
• wald.univariateScore( ... )
• univariateScore.ma( ... )
• IdentifyEquivalence.ma( ... )
• identifyTheEquivalent.ma( ... )
• apply_ideq.ma( ... )
• max_min_assoc.ma( ... )
• min_assoc.ma( ... )
• fs.reg_2( ... )
• gammafsreg_2( ... )
• beta.fsreg_2( ... )
• zip.fsreg_2( ... )
• internaliamb.zipbs( ... )
• internaliamb.betabs( ... )
• iamb.gammabs( ... )
• iamb.normlogbs( ... )
• internaliamb.gammabs( ... )
• internaliamb.normlogbs( ... )
• bic.tobit.fsreg( ... )
• iamb.tobitbs( ... )
• internaliamb.tobitbs( ... )
• ebic.fbed.beta( ... )
• ebic.fbed.cr( ... )
• ebic.fbed.glm( ... )
• ebic.fbed.lm( ... )
• ebic.fbed.mmreg( ... )
• ebic.fbed.multinom( ... )
• ebic.fbed.nb( ... )
• ebic.fbed.ordinal( ... )
• ebic.fbed.tobit( ... )
• ebic.fbed.wr( ... )
• ebic.fbed.zip( ... )
• ebic.beta.bsreg( ... )
• ebic.cr.bsreg( ... )
• ebic.glm.bsreg( ... )
• ebic.lm.bsreg( ... )
• ebic.mm.bsreg( ... )
• ebic.multinom.bsreg( ... )
• ebic.ordinal.bsreg( ... )
• ebic.tobit.bsreg( ... )
• ebic.wr.bsreg( ... )
• ebic.zip.bsreg( ... )
• cat_condis( ... )
• pearson_condis( ... )
• pearson_condis.rob( ... )
• disctor_condis( ... )
• comb_condis( ... )
• fbed.glmm( ... )
• fbed.lmm( ... )
• ebic.fbed.glmm( ... )
• ebic.fbed.lmm( ... )
• lmm.bsreg( ... )
• clogit.fsreg( ... )
• clogit.fsreg_2( ... )
• bic.clogit.fsreg( ... )
• quasibinom.fsreg( ... )
• quasipois.fsreg( ... )
• quasibinom.fsreg_2( ... )
• quasipois.fsreg_2( ... )
• fbed.geeglm( ... )
• fbed.geeglm.reps( ... )
• fbed.ordgee( ... )
• internaliamb.mmbs( ... )
• iamb.gammabs( ... )
• iamb.normlogbs( ... )
• internaliamb.gammabs( ... )
• internaliamb.normlogbs( ... )
• bic.wr.fsreg( ... )
• wr.fsreg( ... )
• wr.fsreg_2( ... )
• ebicScore( ... )
• fbed.glmm.reps( ... )
• fbed.lmm.reps( ... )
• ebic.fbed.lmm.reps( ... )
• ebic.fbed.glmm.reps( ... )
• ebic.glmm.reps.bsreg( ... )
• fbed.ordgee.reps( ... )
• fbed.geelm( ... )
• fbed.geelm.reps( ... )
• univariateScore.gee( ... )
• InternalMMPC.gee( ... )
• InternalSES.gee( ... )
• max_min_assoc.ggee( ... )
• min_assoc.ggee( ... )
• IdentifyEquivalence.ggee( ... )
• identifyTheEquivalent.ggee( ... )
• univariateScore.timeclass( ... )
• InternalMMPC.timeclass( ... )
• gomp2( ... )
• kfbed.reg( ... )
• kfbed.glmm.reg( ... )
• kfbed.gee.reg( ... )
• bs.g2( ... )
• fbed.g2( ... )
• fbed.glmm.ordinal( ... )
• fbed.glmm.ordinal.reps( ... )
• ebic.fbed.glmm.ordinal( ... )
• ebic.fbed.glmm.ordinal.reps( ... )
• glmm.ordinal.bsreg( ... )
• glmm.ordinal.reps.bsreg( ... )
• ebic.glmm.reps.bsreg( ... )
• ebic.glmm.ordinal.reps.bsreg( ... )
• big.model( ... )
• big.fbed.g2( ... )
• big.bs.g2( ... )
• clogit.cv.ses( ... )
• bsreg.big( ... )
• fbed.glmm.cr( ... )
• ebic.fbed.glmm.cr( ... )
• glmm.cr.bsreg( ... )
• ebic.glmm.cr.bsreg( ... )
• test.maker( ... )
• ebic.model( ... )
• fbed.lr( ... )
• fbed.ebic( ... )
• spml.bsreg( ... )
• bic.llr.fsreg( ... )
• ebic.llr.bsreg( ... )
• llr.bsreg( ... )
• beta.reg( ... )
• ebic.spml.bsreg( ... )
• regzinb( ... )
• zinb.mle( ... )
• fbed.glmm.nb( ... )
• fbed.glmm.nb.reps( ... )
• glmm.nb.bsreg( ... )
• glmm.nb.reps.bsreg( ... )
Neighbours of nodes in an undirected graph

Returns the node(s) and their neighbour(s), if there are any.

Description

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

Usage

nei(G, node)

Arguments

G

The adjacency matrix of an undirected graph as produced by \texttt{mmhc.skel}, \texttt{pc.skel} or any other algorithm.

node

A vector with one or more numbers indicating the selected node(s) (or variable(s)).

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: Michail Tsagris <mtsagris@uoc.gr>

See Also

\texttt{mmhc.skel}, \texttt{SES}, \texttt{MMPC}

Examples

# simulate a dataset with continuous data
set.seed(1234)
dataset <- matrix(runif(500 * 20, 1, 100), nrow = 500 )
G <- pc.con(dataset)@G
plotnetwork(G)
Network construction using the partial correlation based forward regression or FBED

```r
dev.new()
nei( G, c(3, 4) )
nei( G, c(1, 3) )
```

Network construction using the partial correlation based forward regression or FBED

**Description**

Network construction using the partial correlation based forward regression or FBED.

**Usage**

```r
corfs.network(x, threshold = 0.05, tolb = 2, tolr = 0.02, stopping = "BIC",
symmetry = TRUE, nc = 1)
corfbed.network(x, threshold = 0.05, symmetry = TRUE, nc = 1)
```

**Arguments**

- `x`: A matrix with continuous data.
- `threshold`: Threshold (suitable values in (0, 1)) for assessing p-values significance. Default value is 0.05.
- `tolb`: The difference in the BIC between two successive values. By default this is set to 2. If for example, the BIC difference between two successive models is less than 2, the process stops and the last variable, even though significant does not enter the model.
- `tolr`: The difference in the adjusted $R^2$ between two successive values. By default this is set to 0.02. If for example, the difference between the adjusted $R^2$ of two successive models is less than 0.02, the process stops and the last variable, even though significant does not enter the model.
- `stopping`: The stopping rule. The "BIC" is the default value, but can change to "ar2" and in this case the adjusted $R^2$ is used. If you want both of these criteria to be satisfied, type "BICR2".
- `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. There was an extra argument for plotting the skeleton but it does not work with the current visualisation packages, hence we removed the argument. Use `plotnetwork` to plot the skeleton.
Details

In the MMHC algorithm (see `mmhc.skel`), the MMPC or SES algorithms are run for every variable. Hence, one can use forward regression for each variable and this is what we are doing here. Partial correlation forward regression is very efficient, since only correlations are being calculated.

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 \( \frac{\text{#edges}}{n(n - 1)/2} \), where \( n \) is the number of variables.

- **info**
  - Some summary statistics about the edges, minimum, maximum, mean, median number of edges.

- **ntests**
  - The number of tests MMPC (or SES) performed at each variable.

- **G**
  - The adjacency 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: Michail Tsagris <mtsagris@uoc.gr>

References


See Also

`mmhc.skel`, `pc.skel`

Examples

```r
# simulate a dataset with continuous data
dataset <- matrix(runif(400 * 20, 1, 100), ncol = 20)
a1 <- mmhc.skel(dataset, max_k = 3, threshold = 0.05, test = "testIndFisher",
               nc = 1)
a2 <- corfs.network(dataset, threshold = 0.05, tolb = 2, tolr = 0.02, stopping = "BIC",
                   symmetry = TRUE, nc = 1)
```
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 performs the 4 orientation rules. A graph is also possible to visualize.

Usage

```r
pc.or(mod)
```

Arguments

- **mod**: An object with the results of the PC algorithm, as produced by `pc.skel` or `pc.con`. There was an extra argument for plotting the skeleton but it does not work with the current visualisation packages, hence we removed the argument. Use `plotnetwork` to plot the skeleton.

Details

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

1. **Rule 0.** 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 1.** 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 2.** 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 3.** 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. If when a rule is applied and a cycle is detected, the rule is cancelled. Also, when applying Rules 1-3 we try to avoid the creation of new v-structures (X -> Y <- Z).
Orientation rules for the PC algorithm

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 adjacency 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@uoc.gr>

References


See Also

- `pc.con`, `pc.skel`, `mmhc.skel`, `is.dag`, `mb`

Examples

```r
# simulate a dataset with continuous data
y <- rdag2(2000, p = 20, nei = 3)
ind <- sample(1:20, 20)
tru <- y$G[ind, ind]
x <- y$x[, ind]
mod <- pc.con(x)
mod$runtime

plotnetwork(tru)

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

plotnetwork( dag2eg(tru) )  # essential graph
plotnetwork(b$G)
```

Partial correlation between two variables

Description

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

Usage

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

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.
- `n`: The sample size of the data from which the correlation matrix was computed.

Details

Given a correlation matrix the function will calculate the partial correlation between variables `indx` and `indy` conditioning on variable(s) `indz`. The logarithm of the p-value is also returned.

Value

The partial correlation coefficient and the logged p-value for the test of no association.

Author(s)

Michail Tsagris

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

See Also

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

Examples

```r
r <- cor( iris[, 1:4] )
partialcor(r, 1, 2, 0, 150)
r[1, 2]  ## the same as above

y <- as.vector( iris[, 1] )
x <- as.vector( iris[, 2] )
z <- as.vector( iris[, 3] )
```
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(x1, x2, R = 999)
permcorrels(y, x, R = 999)

Arguments

x1 A numerical vector.
x2 A numerical vector of the same size as x1.
y A vector whose length is equal to the number of rows of x.
x This is a matrix with many variables.
R The number of permutations to be conducted; set to 999 by default.

Details

This is a computational non parametric (permutation based) 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 logged p-value.

Author(s)

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

References

See Also

pc.skel, testIndSpearman, testIndFisher, SES, CondIndTests

Examples

```r
MXM::permcor(iris[, 1], iris[, 2], R = 999)
x <- matrix(rnorm(50 * 100), ncol = 100)
a <- permcorrels(iris[1:50, 1], x)
```

Description

Plot of longitudinal data.

Usage

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

Arguments

- `target`: A numerical vector with the longitudinal data.
- `tp`: The time points. It can either be a vector with length equal to the number of time points or equal to the length of the target.
- `id`: A numerical vector specifying the subjects. It can either be a vector with length equal to the number of subjects or equal to the length 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.
- `main`: This is a graphical parameter. The title of the graph.

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@uoc.gr>

References


See Also

testIndGLMMReg, SES.glmm

Examples

```r
y <- rnorm(70)
Days <- rep(1:7, each = 10)
id <- rep(1:10, 7)
tc.plot(y, Days, id)
tc.plot(y, Days, id, type = "b")
```

Description

Probability residual of ordinal logistic regression.

Usage

```r
ord.resid(y, est)
```

Arguments

- **y**: An ordered factor variable or a numerical vector.
- **est**: A matrix with the fitted values of an ordinal logistic regression model.

Details

The probability residual of an ordinal logistic regression model is calculated (Li and Shepherd, 2012). It is a vector, irrespective of how many categories there are.
Value

A vector with the probability residuals.

Author(s)

Michail Tsagris

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

References


See Also

testIndOrdinal, ordinal.reg

Examples

library(MASS)
y <- factor(rbinom(400, 3, 0.6), ordered = TRUE)
x <- rnorm(400)
mod <- MASS::polr(y ~ x)
res <- ord.resid(y, mod$fitted.values)
Details

The data (matrix) which will be read and compressed into a big.matrix object must be of type "numeric". We tested it and it works with "integer" as well. But, in general, bear in mind that only matrices will be read. We have not tested with data.frame for example. Whatever, in the help page of "bigmemory" this is mentioned: Any non-numeric entry will be ignored and replaced with NA, so reading something that traditionally would be a data.frame won’t cause an error. A warning is issued. In all cases, the object size is always 696 bytes!

Value

A big.matrix object.

Author(s)

Michail Tsagris

R implementation and documentation: Michail Tsagris <mtsagris@uoc.gr>. For more information see the "bigmemory" package.

See Also

big.gomp

Examples

## Not run:
dataset <- matrix( runif(100 * 50, 1, 100), ncol = 50 )
write.csv(dataset, "dataset.csv", sep = ",")
a <- read.big.data("dataset.csv", header = TRUE)
## End(Not run)
Arguments

**target**
The target (dependent) variable. It can be a numerical variable, factor, ordinal factor, percentages, or time to event.

**dataset**
The predictor variable(s). It can be a vector, a matrix with continuous only variables. If there are no predictor variables leave this NULL.

**test**
Unlike `reg.fit` this accepts the test. The test argument is exactly like in all feature selection methods. This function accepts the following: "testIndReg", "testIndPois", "testIndNB", "testIndLogistic", "testIndMMReg", "testIndRQ", "testIndBinom", "censIndCR", "censIndWR", "censIndLLR", "testIndBeta", "testIndGamma", "testIndNormLog", "testIndTobit", "testIndQPois", "testIndQBinom", "testIndFisher", "testIndMultinom" and "testIndOrdinal".

Details

This is a generic regression function designed for continuous predictor variables only. It was useful for me so I decided to export it.

Value

A list including:

- **mod**
The fitted model.
- **dev**
The deviance. For some models though ("testIndMMReg", "testIndRQ", "censIndCR", "censIndWR", "testIndTobit", "testIndBeta", "testIndNB", "testIndQPois", "testIndQBinom") this contains twice the log-likelihood.
- **bic**
The BIC of the model. This is NA for the "testIndQPois" and "testIndQBinom" because they are quasi likelihood models and hence have no BIC.
- **res**
The residuals of the fitted model.

Author(s)

Michail Tsagris

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

References

Almost the same as in CondIndTests.

See Also

`reg.fit, fbedreg.bic, mmpc.model, ridge.reg`

Examples

```r
# simulate a dataset with continuous data
dataset <- matrix(runif(100 * 5, 1, 100), nrow = 100 )
# the target feature is the last column of the dataset as a vector
target <- dataset[, 1]
```
dataset <- dataset[, -1]
a <- modeler(target, dataset)

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)
waldses.model(target, dataset, wei = NULL, wald.sesObject, nsignat = 1, test = NULL)
waldmmpc.model(target, dataset, wei = NULL, wald.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 or data.frame 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. An example where weights are used is surveys when stratified sampling has occurred.

sesObject
An object with the results of a SES run.

mmpcObject
An object with the results of an MMPC run.

wald.sesObject
An object with the results of a wald.ses run.

wald.mmpcObject
An object with the results of an wald.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 number 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 example.

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@uoc.gr>

References


See Also

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

# simulate a dataset with continuous data
dataset <- matrix( runif(500 * 20, 1, 20), nrow = 500 )
#the target feature is the last column of the dataset as a vector
target <- dataset[, 20]
dataset <- dataset[, -20]

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 based on SES.timeclass and MMPC.timeclass outputs

Regression model(s) obtained from SES.timeclass or MMPC.timeclass

Description

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

Usage

mmpc.timeclass.model(target, dataset, id, reps, wei = NULL, mmpctimeclass.Object)

ses.timeclass.model(target, dataset, id, reps, wei = NULL, sestimeclass.Object,
nsignat = 1)

Arguments

target
The class variable. Provide a vector or a factor with discrete numbers indicating
the class. Its length is equal to the number of rows of the dataset.

dataset
The dataset; provide a matrix. Currently, only continuous datasets are supported.
The dataset contains longitudinal data, where each column is a variable. The
repeated measurements are the samples.
id
A numeric vector containing the subjects or groups. Its length is equal to the
number of rows of the dataset.
reps
A numeric vector containing the time points of the subjects. Its length is equal
to the number of rows of the dataset.
wei
A vector of weights to be used for weighted regression. The default value is
NULL.
mmpctimeclass.Object
An object with the results of an MMPC.timeclass run.

sestimeclass.Object
An object with the results of a SES.timeclass run.

nsignat
How many signatures to use. If nsignat = 1 (default value) the first set of vari-
ables will be used for the model. If you want more, then specify the number
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.
Details
This command is useful if you want to see all models and check for example their fitting ability.

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: Michail Tsagris <mtsagris@uoc.gr>

References

See Also
MMPC.timeclass

Examples
```r
## assume these are longitudinal data, each column is a variable (or feature)
dataset <- matrix( rnorm(400 * 50), ncol = 50 )
id <- rep(1:80, each = 5)  ## 80 subjects
reps <- rep( seq(4, 12, by = 2), 80)  ## 5 time points for each subject
## dataset contains are the regression coefficients of each subject's values on the
## reps (which is assumed to be time in this example)
target <- rep(0:1, each = 200)
a <- MMPC.timeclass(target, reps, id, dataset)
mmpp.timeclass.model(target, dataset, id, reps, mmpctimeclass.Object = a)
```

Description
Generic regression modelling function.
Regression models fitting

Usage

```r
reg.fit(y, dataset, event = NULL, reps = NULL, group = NULL, slopes = FALSE,
        reml = FALSE, model = NULL, 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 \( \log(\text{target}/(1 - \text{target})) \). If they are compositional data the additive log-ratio (multivariate logit) transformation is applied beforehand. If the model is "clogit" this must be a matrix with two columns. The first column 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`: The independent 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 argument 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. Possible values (apart from NULL) are: "gaussian" (default), "binary", "binomial", "multinomial", "poisson", "ordinal", "gamma", "normlog", "tobit", "cox", "weibull", "exponential", "zip", "beta", "median", "negbin", "longitudinal", "grouped", "qpois" and "qbinom". 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. "MM" stands for robust regression using MM estimation and "clogit" stands for conditional logistic regression.

- `wei`: A vector of weights to be used for weighted regression. The default value is NULL. An example where weights are used is surveys when stratified sampling has occurred.

- `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"
Ridge regression

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@uoc.gr>

References
Almost the same as in CondIndTests.

See Also
modeler, fbedreg.bic, mmpc.model, ridge.reg

Examples

#simulate a dataset with continuous data
dataset <- matrix(runif(100 * 5, 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

Description
Regularisation via ridge regression is performed.

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

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>target</td>
<td>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(\text{target}/(1 - \text{target})) ).</td>
</tr>
<tr>
<td>dataset</td>
<td>A numeric matrix containing the variables. Rows are samples and columns are features.</td>
</tr>
<tr>
<td>lambda</td>
<td>The value of the regularisation parameter ( \lambda ).</td>
</tr>
</tbody>
</table>
Ridge regression

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 error 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@uoc.gr>

References


See Also

ridgereg.cv

Examples

#simulate a dataset with continuous data
dataset <- matrix(runif(100 * 30, 1, 100), nrow = 100 )
#the target feature is the last column of the dataset as a vector
target <- dataset[, 10]
dataset <- dataset[, -10]
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

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 \( \lambda \).

Details
For every value of \( \lambda \) the coefficients are obtained. They are plotted versus the \( \lambda \) values.

Value
A plot with the values of the coefficients as a function of \( \lambda \).

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

References

See Also
ridge.reg, ridgereg.cv
Examples

```r
# 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)
```

Description

Receiver operating curve and area under the curve.

Usage

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

Arguments

- `group`: A numerical vector with the predicted values of each group as 0 and 1.
- `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 area under the curve is returned. The user has the option of getting the receiver operating curve as well.

Value

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 pair of 1 - specificity and sensitivity where the Youden’s J appears on the graph and the Youden index which is defined as the maximum value of sensitivity - specificity + 1.
- `auc`: The area under the curve, plus a circle with the point where Youden’s J is located. If "roc" is set to FALSE, this is the only item in the list to be returned.
Search for triangles in an undirected graph

Author(s)

Michail Tsagris

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

See Also

bbc, testIndLogistic

Examples

g <- rbinom(150, 1, 0.6)
f <- rnorm(150)
auc(g, f, roc = FALSE)

Description

Search for triangles in an undirected graph.

Usage

triangles.search(G)

Arguments

G

The adjacency matrix of an undirected graph. G[i, j] = G[j, i] = 1 means there is an edge between modes i and j. Zero values indicate the absence of edges.

Details

The functions searches for triangles, that is for tripletes of nodes (or variables) for which X-Y, Y-Z and X-Z.

Value

A matrix with three columns. If there are no triangles, the matrix is empty. If there is at least one triangle, then each row contains three numbers, one for each node. See the examples below.

Author(s)

Michail Tsagris

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

plotnetwork, pc.skel, mmhc.skel

Examples

```r
set.seed(123)
x <- rdag2(1000, p = 20, nei = 4)$x
a <- pc.skel(x, alpha = 0.05)
plotnetwork(a$G)
triangles.search(a$G)
```

SES.glm.output-class  
Class "SES.glm.output"

Description

SES.glm output object class.

Objects from the Class

Objects can be created by calls of the form `new("SES.glm.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"
- `n.tests`: Object of class "numeric"
- `runtime`: Object of class "proc_time"
- `test`: Object of class "character"
- `correl`: Object of class "character"
- `se`: Object of class "character"

Methods

- `plot`  
  `plot(x = "SES.glm.output", mode = "all")`: Generic function for plotting the generated pvalues of the SES.glm.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.gee, SES.gee

Examples
showClass("SES.glmm.output")

---

**SES.glmm.output-class**  
Class "SES.glmm.output"

Description
SES.glmm output object class.

Objects from the Class
Objects can be created by calls of the form `new("SES.glmm.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"
- `n.tests`: Object of class "numeric"
- `runtime`: Object of class "proc_time"
- `test`: Object of class "character"
- `slope`: Object of class "logical"

Methods
- `plot`: `plot(x = "SES.glmm.output", mode = "all")`: Generic function for plotting the generated pvalues of the SES.glmm.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.glmm, SES.glmm

Examples

```r
showClass("SES.glmm.output")
```

## SESoutput-class

### 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"
- `n.tests`: Object of class "numeric"
- `runtime`: Object of class "proc_time"
- `test`: Object of class "character"

### Methods

- `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.
Skeleton (local) around a node of the max-min hill-climbing (MMHC) algorithm

Author(s)
Giorgos Athineou <athineou@csd.uoc.gr>

See Also
SES

Examples

showClass("SESoutput")

---

Skeleton (local) around a node of the max-min hill-climbing (MMHC) algorithm

Description

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

Usage

local.mmhc.skel(dataset, node, max_k = 3, threshold = 0.05, test = "testIndFisher")

Arguments

dataset A matrix with the variables. The user must know if they are continuous or if they are categorical. If you have a matrix with categorical data, i.e. 0, 1, 2, 3 where each number indicates a category, the minimum number for each variable must be 0. data.frame is also supported, as the dataset in this case is converted into a matrix.

node A number between 1 and the number of columns of the dataset. The local network (edges only) will be built around this node. At first the parents and children of this node are identified and then their parents and children. No inconsistencies correction whatsoever is attempted. A variable detected by the node, but the node was not detected by that variable.

max_k The maximum conditioning set to use in the conditional independence 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. Or in general, if the dataset is a data.frame with different types of data, leave this NULL. See also link(MMPC) for the automatic choice of tests.
Skeleton (local) around a node of the max-min hill-climbing (MMHC) algorithm

Details

The MMPC is run on the user specific variable. The backward phase (see Tsamardinos et al., 2006) takes place automatically. Then, the MMPC is run on the parents and children of that variable. If the node variable is not detected by a variable, this variable is not removed though.

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.
- **ntests**: The number of tests MMPC (or SES) performed at each variable.
- **res**: A list with the parents and children of each variable. The first element is the parents and children of the node variable.
- **Gloc**: The local adjacency matrix. A value of 1 in $G[i, j]$ may not appear in $G[j, i]$ also, indicating that variable $j$ was discovered as a possible parent or child of node $i$, but not the converse. The usual MMHC (mmhc.skel) removes the edge between them as this is an inconsistency.

Author(s)

Michail Tsagris

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

References


See Also

mmhc.skel, pc.skel, pc.or, corfs.network

Examples

# simulate a dataset with continuous data
dataset <- matrix(runif(500 * 30, 1, 100), nrow = 500)
a1 <- mmhc.skel(dataset, max_k = 3, threshold = 0.05, test = "testIndFisher")
a2 <- local.mmhc.skel(dataset, 4)
a1$runtime
da2$runtime

dataset <- rdag2(500, p = 20, nei = 3)$x
a1 <- mmhc.skel(dataset, max_k = 3, threshold = 0.05, test = "testIndFisher")
a2 <- local.mmhc.skel(dataset, 5)
a1$runtime
da2$runtime
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

```r
mmhc.skel(dataset, max_k = 3, threshold = 0.05, test = "testIndFisher", type = "MMPC",
hash = FALSE, backward = TRUE, symmetry = TRUE, nc = 1, ini.pvalue = NULL)
```

```r
glmm.mmhc.skel(dataset, group, max_k = 3, threshold = 0.05, test = "testIndLMM",
type = "MMPC.glmm", hash = FALSE, symmetry = TRUE, nc = 1, ini.pvalue = NULL)
```

```r
gee.mmhc.skel(dataset, group, max_k = 3, threshold = 0.05, test = "testIndGEEReg",
type = "MMPC.gee", se = "jack", hash = FALSE, symmetry = TRUE, nc = 1,
inip.value = NULL)
```

Arguments

- **dataset** A matrix with the variables. The user must know if they are continuous or if they are categorical. If you have a matrix with categorical data, i.e. 0, 1, 2, 3 where each number indicates a category, the minimum number for each variable must be 0. For the "glmm.mmhc.skel" this must be only a matrix.

- **group** This is to be used in the "glmm.mmhc.skel" and "gee.mmhc.skel" only. It is a vector for identifying the grouped data, the correlated observations, the subjects.

- **max_k** The maximum conditioning set to use in the conditional independence 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. In case the dataset is a data.frame with mixed types of data leave this NULL and an appropriate test will be selected. See MMPC for the automatic choice of tests. For the "glmm.mmhc.skel" the available tests in MMPC.glmm.

- **type** The type of variable selection to take place for each variable (or node). The default (and standard) is "MMPC" and "MMPC.glmm". You can also choose to run it via "SES" and "SES.glmm" and thus allow for multiple signatures of variables to be connected to a variable.

- **se** The method for estimating standard errors. This is very important and crucial. The available options for Gaussian, Logistic, Poisson and Gamma regression
are: a) 'san.se': the usual robust estimate, b) 'jack': approximate jackknife variance estimate. c) 'j1s': if 1-step jackknife variance estimate and d) 'fij': fully iterated jackknife variance estimate. If you have many clusters (sets of repeated measurements) "san.se" is fine as it is asymptotically correct, plus jacknife estimates will take longer. If you have a few clusters, then maybe it’s better to use jacknife estimates.

The jackknife variance estimator was suggested by Paik (1988), which is quite suitable for cases when the number of subjects is small (K < 30), as in many biological studies. The simulation studies conducted by Ziegler et al. (2000) and Yan and Fine (2004) showed that the approximate jackknife estimates are in many cases in good agreement with the fully iterated ones.

hash
A boolean variable which indicates whether (TRUE) or not (FALSE). Default value is FALSE. If TRUE a hashObject is produced and hence more memory is required. If you want to know the number of tests executed for each variable then make it TRUE.

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. It call the mmpcbackphase for this purpose. For perm.ses and wald.ses this is not yet applicable.

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. There was an extra argument for plotting the skeleton but it does not work with the current visualisation packages, hence we removed the argument. Use plotnetwork to plot the skeleton.

ini.pvalue
This is a list with the matrix of the univariate p-values. If you want to run mmhc.skel again, the univariate associations need not be calculated again.

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 \( \frac{\text{#edges}}{n(n-1)/2} \), where \( n \) is the number of variables.

- **info**: Some summary statistics about the edges, minimum, maximum, mean, median number of edges.

- **ms**: If you run "MMPC" for each variable this is NULL. If you run "SES" is a vector denoting which variables had more than one signature, i.e. more than one set of variables associated with them.

- **ntests**: The number of tests MMPC (or SES) performed at each variable.

- **ini.pvalue**: A matrix with the p-values of all pairwise univariate associations.

- **pvalue**: A matrix with the final p-values. These are the maximum p-values calculated during the process. When the process finishes, the matrix is not symmetric. It becomes symmetric though by keeping the maximum between any two off-diagonal elements. These p-values now can be used to sort the strength of the edges. If you know the true adjacency matrix you can use them and create a ROC curve (see `bn.skel.utils` for more information).

- **G**: The adjacency 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: Michail Tsagris <mtsagris@uoc.gr>

References


Skeleton of the PC algorithm


See Also

pc.skell, pc.or, corfs.network, bn.skel.utils

Examples

# simulate a dataset with continuous data
y <- rdag2(500, p = 20, nei = 3)
x <- y$x
a <- mmhc.skel(x, max.k = 5, threshold = 0.01, test = "testIndFisher"
)b <- pc.skel( x, alpha = 0.01 )
a$runtime
b$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 for continuous data only and it calls the same C++ as pc.skel. Hence, you are advised to use pc.skel.

Usage

pc.skell(dataset, method = "pearson", alpha = 0.01, rob = FALSE, R = 1, stat = NULL, ini.pvalue = NULL)

pc.con(dataset, method = "pearson", alpha = 0.01)

pc.skell.boot(dataset, method = "pearson", alpha = 0.01, R = 199, ncores = 1)

glm.pc.skell(dataset, group, method = "comb.mm", alpha = 0.01, stat = NULL, ini.pvalue = NULL)

glm.pc.skell(dataset, group, se = "jack", method = "comb.mm", alpha = 0.01, stat = NULL, ini.pvalue = NULL)
Arguments

**dataset**  
A matrix with the variables. The user must know if they are continuous or if they are categorical. **If you have categorical data though, the user must transform the data.frame into a matrix. In addition, the numerical matrix must have values starting from 0. For example, 0, 1, 2, instead of "A", "B" and "C".** In the case of mixed variables, continuous, binary and ordinal this must a data.frame and the non continuous variables must be ordered factors, even the binary variables.

For the pc.con, pc.skel.boot and glmm.pc.skel, the dataset can only be a matrix.

**method**  
If you have continuous data, you can choose either "pearson", "spearman" or "distcor". The latter uses the distance correlation and should not be used with lots of observations as it is by default really slow. If you have categorical data, this must be "cat". If you have a mix of continuous, binary and ordinal data (we will expand the available dataset in the future) then choose "comb.fast" or "comb.mm". These two methods perform the symmetric test for mixed data (Tsagris et al., 2018). See details for more information on this. For the mixed models PC algorithm, this is the same argument, but currently only "comb.mm" is accepted.

**group**  
This is to be used in the "glmm.pc.skel" and "gee.pc.skel" only. It is a vector for identifying the grouped data, the correlated observations, the subjects.

**se**  
The method for estimating standard errors. This is very important and crucial. The available options for Gaussian, Logistic, Poisson and Gamma regression are: a) 'san.se': the usual robust estimate, b) 'jack': approximate jackknife variance estimate. c) 'j1s': if 1-step jackknife variance estimate and d) 'fij': fully iterated jackknife variance estimate. If you have many clusters (sets of repeated measurements) "san.se" is fine as it is asymptotically correct, plus jacknife estimates will take longer. If you have a few clusters, then maybe it's better to use jacknife estimates.

The jacknife variance estimator was suggested by Paik (1988), which is quite suitable for cases when the number of subjects is small (K < 30), as in many biological studies. The simulation studies conducted by Ziegler et al. (2000) and Yan and Fine (2004) showed that the approximate jacknife estimates are in many cases in good agreement with the fully iterated ones.

**alpha**  
The significance level ( suitable values in (0, 1) ) for assessing the p-values. Default value is 0.01.

**rob**  
This is for robust estimation of the Pearson correlation coefficient. Default value is FALSE.

**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. There was an extra argument for plotting the skeleton but it does not work with the current visualisation packages, hence we removed the argument. Use plotnetwork to plot the skeleton.

In the pc.skel.boot this is the number of bootstrap resamples to draw. The PC algorithm is performed in each bootstrap sample. In the end, the adjacency matrix on the observed data is returned, along with another adjacency matrix.
Skeleton of the PC algorithm

produced by the bootstrap. The latter one contains values from 0 to 1 indicating the proportion of times an edge between two nodes was present.

**ncores**  The number of cores to use. By default this is set to 1.

**stat**  If you have the initial test statistics (univariate associations) values supply them here.

**ini.pvalue**  If you have the initial p-values of the univariate associations supply them here.

**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 (Spirtes et al., 2000, pg. 90). At every step of the algorithm use the pairs which are least statistically associated. The conditioning set consists of variables which are most statistically associated with each other of the pair of variables.

For example, for the pair (X, Y) there can be two conditioning 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. In the event of 2 or more p-values being the same (with permutations for example), 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. 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 Tsamardinos et al., 2006).

The "comb.fast" and "comb.mm" methods are used with mixed variables, continuous, binary and ordinal. The "comb.mm" performs two log-likelihood ratio tests. For every pair of variables each of the two variables is treated as response and the suitable regression model is fitted. Then, two likelihood ratio tests are performed and the 2 p-values are combined in a meta-analytic way. In the case of "comb.fast" one regression model is fitted, the easiest (between the two) to implement. The ordering of the "easiness" is as follows: linear regression > logistic regression > ordinal regression.

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.

The "glmm.pc.skel" and "gee.pc.skel" are designed for clustered or grouped data. It uses linear mixed models and works in the same way as the PC with mixed data. For each variable, a random intercepts model is fitted and the significance of the other variable is assessed. The two p-values are meta-analytically combined.

For all cases, we return the maximum logged p-value of the conditional independence tests between the pairs. This can be used to order the strength of association between pairs of variables. In addition, one can use it to estimate the AUC. See the example in `bn.skel.utils`

If you want to use the GEE methodology, make sure you load the library geepack first.
Skeleton of the PC algorithm

Value

A list including:

- `stat`: The test statistics of the univariate associations.
- `ini.pvalue`: The initial univariate associations p-values.
- `pvalue`: The logarithm of the maximum p-values from every conditional independence test. Note also, that if there is a log p-value smaller that the log of the threshold value that means there is an edge. This can be used to estimate the FDR (Tsamardinos and Brown, 2008). See details for more information. This is also an estimate of the strength of the association between pairs of variables.
- `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 \( \frac{\text{#edges}}{n(n - 1)/2} \), where \( n \) is the number of variables.
- `info`: Some summary statistics about the edges, minimum, maximum, mean, median number of edges.
- `G`: The adjacency 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@uoc.gr>.

References


See Also

bn.skel.utils, mmhc.skel, corfs.network, local.mmhc.skel

Examples

# simulate a dataset with continuous data
y <- rdag2(300, p = 20, nei = 3)
ind <- sample(1:20, 20)
x <- y$x[, ind]
a <- mmhc.skel(x, max_k = 3, threshold = 0.05, test = "testIndFisher" )
b <- pc.skel(x, method = "pearson", alpha = 0.05 )
a$runtime
b$runtime

Description

Structural Hamming distance between two partially oriented DAGs.

Usage

shd(est, true)
**Arguments**

- `est`: The first (partially oriented) DAG. This could also be the estimated DAG.
- `true`: The second (partially oriented) DAG. This could also be the equivalence class of the true DAG.

**Details**

The structural Hamming distance as proposed by Tsamardinos et al. (2006) is calculated and returned. The cases are listed below:

<table>
<thead>
<tr>
<th>True</th>
<th>Estimated</th>
<th>Penalty</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>-</td>
<td>1</td>
</tr>
<tr>
<td>&gt;</td>
<td>&lt;</td>
<td>1</td>
</tr>
<tr>
<td>&lt;</td>
<td>&gt;</td>
<td>1</td>
</tr>
<tr>
<td>-&gt;</td>
<td>&lt;-</td>
<td>1</td>
</tr>
<tr>
<td>&lt;-</td>
<td>-&gt;</td>
<td>1</td>
</tr>
</tbody>
</table>

**Value**

A list including:

- `mat`: A table with the agreements and disagreements between the two DAGs.
- `shd`: The structural Hamming distance.

**Author(s)**

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

**References**


**See Also**

- `pc.skel`, `pc.or`, `mmhc.skel`, `plotnetwork`

**Examples**

```r
y <- rdag(1000, 20, 0.2)
tru <- y$G
mod <- pc.skel(y$x)
a <- pc.or(mod)
shd(a$G, dag2eg(tru))
```
Supervised PCA

Description

Supervised PCA.

Usage

supervised.pca(target, dataset, indices, center = TRUE, scale = TRUE, colours = NULL, graph = TRUE)

Arguments

target A numerical vector or a factor denoting the class of each sample, the response variable.
dataset A matrix with numerical data (the predictor variables).
indices A vector with indices denoting which variables have been selected.
center In the calculation of the PCA, should the data be centered? Default value is TRUE.
scale In the calculation of the PCA, should the data be scaled to unity variance? Default value is TRUE.
colours Should the colour of the points be defined by the target variable or do you want to pass your own colours? This must be a vector whose length is equal to the length of the target.
graph Should two graphs be returned? The scores of the first two principal components based on all the data and based on the selected variables.

Details

This is not exactly the standard supervised PCA as suggested by Bair et al (2006). What we do here essentially is the following: PCA on all variables and on the variables selected by a variable selection algorithm.

Value

A list including:

mod.all The output returned by prcomp applied to all variables.
mode.sel The output returned by prcomp applied to the selected variables.
var.percent The percentage of variance explained by the selected variables.

Author(s)

Michail Tsagris
R implementation and documentation: Michail Tsagris <mtsagris@uoc.gr>
Symmetric conditional independence test with clustered data

References


See Also

gomp, fbed.reg, MMPC

Examples

```r
x <- as.matrix(iris[, 1:4])
target <- iris[, 5]
supervised.pca(target, x, indices = 1:2)
```

Description

Symmetric conditional independence test with clustered data.

Usage

```r
glmm.ci.mm(ind1, ind2, cs = NULL, dat, group)
gee.ci.mm(ind1, ind2, cs = NULL, dat, group, se = "jack")
```

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). If you have no variables set this equal to 0.
- `dat`: A numerical matrix with data.
- `group`: This is a numerical vector denoting the groups or the subjects.
- `se`: The method for estimating standard errors. This is very important and crucial. The available options for Gaussian, Logistic and Poisson regression are: a) `"san.se"`: the usual robust estimate. b) `"jack"`: approximate jackknife variance estimate. c) `"j1s"`: if 1-step jackknife variance estimate and d) `"fij"`: fully iterated jackknife variance estimate. If you have many clusters (sets of repeated measurements) "san.se" is fine as it is asymptotically correct, plus jackknife estimates will take longer. If you have a few clusters, then maybe it’s better to use jackknife estimates.

The jackknife variance estimator was suggested by Paik (1988), which is quite suitable for cases when the number of subjects is small (K < 30), as in many biological studies. The simulation studies conducted by Ziegler et al. (2000) and Yan and Fine (2004) showed that the approximate jackknife estimates are in many cases in good agreement with the fully iterated ones.
Symmetric conditional independence test with clustered data

Details

Two linear random intercept models are fitted, one for each variable and the p-value of the hypothesis test that the other variable is significant is calculated. These two p-values are combined in a meta-analytic way. The models fitted are either linear, logistic and Poisson regression.

Value

A vector including the test statistic, its associated p-value and the relevant degrees of freedom.

Author(s)

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

References


See Also

pc.skel, condi, testIndGLMMReg

Examples

```r
## we generate two independent vectors of clustered data
s1 <- matrix(1.5, 4, 4)
diag(s1) <- 2.5
s2 <- matrix(1.5, 4, 4)
diag(s2) <- 2
x1 <- MASS::mvrnorm(10, rnorm(4), s1)
x1 <- as.vector( t(x1) )
x2 <- MASS::mvrnorm(10, rnorm(4), s2)
x2 <- as.vector( t(x2) )
id <- rep(1:10, each = 4)
glmm.ci.mm(1, 2, dat = cbind(x1,x2), group = id)
gee.ci.mm(1, 2, dat = cbind(x1,x2), group = id)
```
Symmetric conditional independence test with mixed data

Description

Symmetric conditional independence test with mixed data.

Usage

```r
ci.mm(ind1, ind2, cs = NULL, dat, type, rob = FALSE, R = 1)
ci.fast(ind1, ind2, cs = NULL, dat, type, rob = FALSE, R = 1)
ci.mm2(ind1, ind2, cs = NULL, suffStat)
ci.fast2(ind1, ind2, cs = NULL, suffStat)
```

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). If you have no variables set this equal to 0.
- **dat**: A data.frame with numerical, binary, nominal and ordinal variables only.
- **type**: This is obsolete basically, but we need it here, so that the functions `ci.mm` and `ci.fast` have the same signatures as in `cat.ci`, `condi` and `dist.condi`.
- **rob**: This is obsolete basically, but we need it here, so that the functions `ci.mm` and `ci.fast` have the same signatures as in `cat.ci`, `condi` and `dist.condi`.
- **R**: This is obsolete basically, but we need it here, so that the functions `ci.mm` and `ci.fast` have the same signatures as in `cat.ci`, `condi` and `dist.condi`.
- **suffStat**: This is a list with only the dataset and the name must be "dataset". A data.frame with numerical, binary, nominal and ordinal variables only.

Details

The functions "ci.mm" and "ci.fast" are general functions to be used anywhere. The functions "ci.mm2" and "ci.fast2" are designed to be accepted by the command "pc" in the package "pcalg". The functions "ci.mm2" and "ci.fast2" can be fed in the "pc" function of the "pcalg" package in order to produce a PDAG with mixed data using the PC algorithm. For more information see the relevant paper in the references.

The "ci.mm" and "ci.fast" work with linear, logistic, multinomial and ordinal regression, whereas the "ci.mm2" and "ci.fast2" work with linear, logistic and ordinal regression only.

Value

A vector including the test statistic, it’s associated p-value and the relevant degrees of freedom.
The max-min Markov blanket algorithm

Author(s)

Michail Tsagris

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

References

Tsagris M. (2019). Bayesian network learning with the PC algorithm: an improved and correct


Cambridge, MA, USA, 3rd edition.


See Also

pc.skel, mmhc.skel, cat.ci, condi

Examples

```r
CI.mm(1, 2, dat = iris)
CI.mm(1, 5, dat = iris)
CI.fast(1, 5, dat = iris)

x <- iris
x[, 5] <- as.numeric(x[, 5])  ## Caution:: this will be treated as ordered variable.

x[, 5] <- factor(x[, 5], ordered = TRUE)

## ci.mm2 and ci.fast2 do not perform multinomial regression.
CI.mm2(1, 5, suffStat = list(dataset = x) )
CI.fast2(1, 5, suffStat = list(dataset = x) )
```

The max-min Markov blanket algorithm

Max-min Markov blanket algorithm

Description

The MMB 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

```r
mmmb(target, dataset, max_k = 3, threshold = 0.05, test = "testIndFisher",
user_test = NULL, ncores = 1)
```
The max-min Markov blanket algorithm

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 available, 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.

**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

The idea is to run the MMPC algorithm at first and identify the parents and children (PC) of the target variable. As a second step, the MMPC algorithm is run on the discovered variables to return PC. 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 continuous, 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 an 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@uoc.gr>
References


See Also

CondIndTests, MMPC, SES

Examples

```r
set.seed(123)
#simulate a dataset with continuous data
dataset <- matrix( runif(200 * 50, 1, 100), ncol = 50 )
#define a simulated class variable
target <- 3 * dataset[, 10] + 2 * dataset[, 50] + 3 * dataset[, 20] + rnorm(200, 0, 5)
a1 <- mmb(target , dataset , max_k = 3 , threshold = 0.05, test= "testIndFisher",
ncores = 1,)
a2 <- MMPC(target, dataset, test="testIndFisher")
```

Topological sort of a DAG

Description

Topological sort of a DAG.

Usage

topological_sort(dag)

Arguments

dag A square matrix representing a directed graph which contains 0s and 1s. If G[i, j] = 1 it means there is an arrow from node i to node j. When there is no edge between nodes i and j if G[i, j] = 0.

Details

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

Value

A vector with numbers indicating the sorting. If the matrix does not correspond to a DAG, NA will be returned.
Total causal effect of a node on another node

Author(s)

Ioannis Tsamardinos and Michail Tsagris

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

References


See Also

plotnetwork, nei, pc.or

Examples

# simulate a dataset with continuous data
G = rdag(100, 10, 0.3)$G
G[G == 2] <- 1
G[G == 3] <- 0
topological_sort(G)

Description

Total causal effect of a node on another node.

Usage

ida(x, y, G, dataset)

Arguments

x A number between 1 and the number of variables (nodes) indicating the node whose total causal effect we want to estimate. This is the independent variable. See details for more on this.

y A number between 1 and the number of variables (nodes) indicating the node who is the dependent variable. The goal is to estimate the total causal effect of x on y.

G A square matrix representing a (partially) directed graph which contains 0s and 1s. If G[i, j] = 2 it means there is an arrow from node i to node j. If G[i, j] = 1, there is an undirected edge between nodes i and j and there is no edge between nodes i and j if G[i, j] = 0.
dataset

The dataset. This is a numerical matrix with data.

Details

The total causal effect defined in Pearl’s do-calculus is \( E(Y|do(X=z+1)) - E(Y|do(X=z)) \). As Pearl described it, he used linear regression, hence this function works for continuous data which are assumed to be Gaussian.

We estimate a set of possible total causal effects using linear regression. If \( y \) is a parent, or a descendant, of \( x \) in \( G \), the estimated causal effect of \( x \) on \( y \) is zero. If \( y \) is not a parent of \( x \), we take the regression coefficient of \( x \) in the regression \( \text{lm}(y \sim x + \text{pa}(x)) \), where \( \text{pa}(x) \) denotes the parents of \( x \). This is repeated using all parents of \( x \) (including the empty set) and all possible parents values of \( x \) and their combinations.

One restriction to bear in mind. If a collider is created that combination of nodes is not used in the regression.

Value

A list including:

tc

A matrix with 4 elements. The first columns one is the estimated beta coefficient, the second is its standard error, its t-value and the p-value for testing whether this is equal to 0.

mess

If the \( x \) node has no parents a message about this appears. Otherwise this is NULL.

Author(s)

Michail Tsagris

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

References


See Also

plotnetwork, nei, pc.or

Examples

dataset <- rdag2(1000, p = 20, nei = 3)$x
mod <- pc.skel(dataset ,alpha = 0.01)
G <- pc.or(mod)$G
ida(10, 15, G, dataset)
Transformation of a DAG into an essential graph

**Description**

Transforms a DAG into an essential graph.

**Usage**

\[ \text{dag2eg}(\text{dag}, \text{type} = \text{NULL}) \]

**Arguments**

- **dag**
  The graph matrix as produced from \texttt{pc.or} or any other algorithm which produces directed graphs. A DAG in general.

- **type**
  This can be either \texttt{NULL} or 1 or 2. \texttt{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 \texttt{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 \texttt{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: Ioannis Tsamardinos <tsamard@csd.uoc.gr> and and Michail Tsagris <mtsagris@uoc.gr>

**References**


**See Also**

\texttt{plotnetwork, is.dag, topological_sort}
Transitive closure of an adjacency matrix

Examples

```r
# 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 transitive closure of the adjacency matrix representing a graph.

Author(s)

Anna Roumpelaki

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

References

Examples

```r
# example adjacency matrix
# simulate a dataset with continuous data
dataset <- matrix( runif(300 * 20, 1, 100), nrow = 300 )
test <- pc.con( dataset, method = "pearson", alpha = 0.05 )$G
transitiveClosure(test)
```

Description

Undirected path(s) between two nodes.

Usage

```r
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@uoc.gr>

References

Univariate regression based tests

See Also

SES, MMPC, pc.skel

Examples

# simulate a dataset with continuous data
set.seed(1234)
dataset <- matrix(runif(1000 * 10, 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, targetID = -1, test = NULL, user_test = NULL,
wei = NULL, ncores = 1)

ebic.univregs(target, dataset, targetID = -1, test = NULL, user_test = NULL,
wei = NULL, ncores = 1, gam = NULL)

wald.univregs(target, dataset, targetID = -1, test = NULL, user_test = NULL,
wei = NULL, ncores = 1)

perm.univregs(target, dataset, targetID = -1, test = NULL, user_test = NULL,
wei = NULL, threshold = 0.05, R = 999, ncores = 1)

score.univregs(target, dataset, test)

big.score.univregs(target = NULL, dataset, test)

rint.regs(target, dataset, targetID = -1, id, reps = NULL, tol = 1e-07)

glmm.univregs(target, reps = NULL, id, dataset, targetID = -1, test, wei = NULL,
slopes = FALSE, ncores = 1)

ggee.univregs(target, reps = NULL, id, dataset, targetID = -1, test, wei = NULL,
correl = "exchangeable", se = "jack", ncores = 1)
Arguments

target
The target (dependent) variable. It must be a numerical vector. In the case of "big.score.univregs" this can also be NULL if it is included in the first column of the dataset.

dataset
The independent variable(s). For the "univregs" this can be a matrix or a dataframe with continuous only variables, a data frame with mixed or only categorical variables. For the "wald.univregs", "perm.univregs", "score.univregs" and "rint.regs" this can only by a numerical matrix. For the "big.score.univregs" this is a big.matrix object with continuous data only.

targetID
This is by default -1. If the target is not a variable but is in included in the dataset, you can specify the column of dataset playing the role of the target.

test
For the "univregs" this can only be one of the following: testIndFisher, testIndSpearman, gSquare, testIndBeta, testIndLogistic, testIndMultinom, testIndOrdinal, testIndPois, testIndZIP, testIndNB, testIndClogit, testIndBinom, testIndGreg, censIndCR, censIndWR, censIndER, censIndLLR, testIndTobit, testIndGamma, testIndNormLog or testIndSPML for a circular target. For the testIndSpearman the user must supply the ranked target and ranked dataset. The reason for this, is because this function is called internally by SES and MMPC, the ranks have already been applied and there is no reason to re-rank the data.

For the "ebic.univregs" this can only be one of the following: testIndFisher, testIndBeta, testIndReg, testIndLogistic, testIndMultinom, testIndOrdinal, testIndPois, testIndZIP, testIndNB, testIndClogit, testIndBinom, censIndCR, censIndWR, censIndER, censIndLLR, testIndTobit, testIndGamma, testIndNormLog or testIndSPML for a circular target.

For the "wald.univregs" this can only be one of the following: waldBeta, waldCR, waldWR, waldER, waldLLR, waldTobit, waldClogit, waldLogistic, waldPois, waldNB, waldBinom, waldZIP, waldMMReg, waldIGreg, waldOrdinal, waldGamma or waldNormLog.

For the "perm.univregs" this can only be one of the following: permFisher, permReg, permRQ, permBeta, permCR, permWR, permER, permLLR, permTobit, permClogit, permLogistic, permPois, permNB, permBinom, permSquare, permZIP, permMVreg, permIGreg, permGamma or permNormLog.

For the "score.univregs" and "big.score.univregs" this can only be one of the following: testIndBeta, testIndLogistic, testIndMultinom, testIndPois, testIndNB, testIndGamma or censIndWR but with no censored values and simply the vector, not a Surv object. Ordinal regression is not supported.

For the "glmm.univregs" this can only be one of the following: testIndGLMMReg, testIndLMM, testIndGLMMLogistic, testIndGLMМОrdinal, testIndGLMMPois, testIndGLMMNB, testIndGLMMGamma, testIndGLMMNormLog or testIndGLMMC.

For the "gee.univregs" this can only be one of the following: testIndGLMMReg, testIndGLMMLogistic, testIndGLMМОrdinal, testIndGLMMPois, testIndGLMM Gamma or testIndGLMMNormLog.

Note that in all cases you must give the name of the test, without " ".
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.

wei  
A vector of weights to be used for weighted regression. The default value is NULL. An example where weights are used is surveys when stratified sampling has occurred.

gam  
This is for ebic.univregs only and it refers to the gamma value of the eBIC. If it is NULL, the default value is calculated and if gam is zero, the usual BIC is returned. see Chen and Chen (2008) for more information.

threshold  
Threshold (suitable values in (0, 1)) for assessing p-values significance. Default value is 0.05. The reason for this is to speed up the computations. The permutation p-value is calculated as the proportion of times the permuted test statistic is higher than the observed test statistic. When running the permutations, if the proportion is more than 0.05, the process stops. A decision must be made fast, and if the non rejection decision has been made, there is no need to perform the rest permutations; the decision cannot change.

R  
This is for the permutations based regression models. It is the number of permutations to apply. Note, that not all the number of permutations need be performed. If the number of times the test statistic is greater than the observed test statistic is more than threshold * R, the iterations stop, as a decision has already been made.

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.

id  
A numerical vector of the same length as target with integer valued numbers, such as 1, 2, 3,... (zeros, negative values and factors are not allowed) specifying the clusters or subjects. This argument is for the rint.regs (see details for more information).

reps  
If you have measurements over time (logitudinal data) you can put the time here (the length must be equal to the length of the target) or set it equal to NULL. (see details for more information).

tol  
The tolerance value to terminate the Newton-Raphson algorithm in the random effects models.

slopes  
Should random slopes for the ime effect be fitted as well? Default value is FALSE.

correl  
The correlation structure. For the Gaussian, Logistic, Poisson and Gamma regression this can be either "exchangeable" (compound symmetry, suitable for clustered data) or "ar1" (AR(1) model, suitable for longitudinal data). For the ordinal logistic regression its only the "exchangeable" correlation structure.

se  
The method for estimating standard errors. This is very important and crucial. The available options for Gaussian, Logistic, Poisson and Gamma regression
Univariate regression based tests

are: a) 'san.se', the usual robust estimate. b) 'jack': if approximate jackknife variance estimate should be computed. c) 'j1s': if 1-step jackknife variance estimate should be computed and d) 'fij': logical indicating if fully iterated jackknife variance estimate should be computed. If you have many clusters (sets of repeated measurements) "san.se" is fine as it is asymptotically correct, plus jackknife estimates will take longer. If you have a few clusters, then maybe it's better to use jackknife estimates.

The jackknife variance estimator was suggested by Paik (1988), which is quite suitable for cases when the number of subjects is small (K < 30), as in many biological studies. The simulation studies conducted by Ziegler et al. (2000) and Yan and Fine (2004) showed that the approximate jackknife estimates are in many cases in good agreement with the fully iterated ones. This is obsolete for "testIndGEEOrdinal", but is here for compatibility reasons.

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.

For the score.univregs, score based tests are used which are extremely fast.

For the rint.regs, we perform linear mixed models (weights are not allowed) with random intercepts only (no random slopes). This function works for clustered or longitudinal data. The covariance structure we impose is compound symmetry, hence for longitudinal data, this may not be the best option, yet it will work.

If you want to use the GEE methodology, make sure you load the library geepack first.

Value

In the case of "ebic.univregs" a list with one element

ebic The eBlc of every model. If the i-th value is "Inf" it means that the i-th variable had zero variance. It had the same value everywhere.

For all other cases a list including:

stat The value of the test statistic. If the i-th value is zero (0) it means that the i-th variable had zero variance. It had the same value everywhere.

pvalue The logarithm of the p-value of the test. If the i-th value is zero (0) it means that the i-th variable had zero variance. It had the same value everywhere.

Author(s)

Michail Tsagris

R implementation and documentation: Michail Tsagris <mtsagris@uoc.gr>
Utilities for the skeleton of a (Bayesian) Network

References


See Also

cond.regs, SES, MMPC, CondIndTests, reg.fit, ridge.reg

Examples

y <- rpois(50, 15)
x <- matrix(rnorm(50 * 7), ncol = 7)
a1 <- univregs(y, x, test = testIndPois)
a2 <- perm.univregs(y, x, test = permPois)
a3 <- wald.univregs(y, x, test = waldPois)

Utilities for the skeleton of a (Bayesian) Network

Utilities for the skeleton of a (Bayesian) Network

Description

Utilities for the skeleton of a (Bayesian) Network

Usage

bn.skel.utils(mod, G = NULL, roc = TRUE, alpha = 0.01)
bn.skel.utils2(mod, G = NULL, roc = TRUE, alpha = 0.01)

Arguments

mod An object as returned by pc.skel, glmm.pc.skel or mmhc.skel.
G The true adjacency matrix with 1 indicating an edge and zero its absence. Symmetric or not is not important. If this is not available, leave it NULL.
roc Do you want a graph with the ROC curve be returned? Default value is TRUE.
alpha The significance level (suitable values in (0, 1)) for assessing the p-values. Default value is 0.01.
Details

Given the true adjacency matrix one can evaluate the estimated adjacency matrix, skeleton, of the PC or the MMHC algorithm.

The bn.skels.utils give you the area under the curve, false discovery rate and sorting of the edges based on their p-values.

The bn.skel.utils2 estimates the confidence of each edge. The estimated proportion of null p-values is estimated the algorithm by Storey and Tibshirani (2003).

Value

For the "bn.skel.utils" a list including:

- **fdr** The false discovery rate as estimated using the Benjamini-Hochberg correction.
- **area** This is a list with the elements of the auc function. The area under the curve, the sensitivity and specificity for a range of values, the Youden index, etc.
- **sig.pvalues** A matrix with the row and column of each significant p-value sorted in ascending order. As we move down the matrix, the p-values increase and hence the strength of the associations decreases.

For the "bn.skel.utils2" a list including:

- **area** This is a list with the elements of the auc function. The area under the curve, the sensitivity and specificity for a range of values, the Youden index, etc.
- **pxy** A matrix with the row and column of the confidence of each p-value sorted in ascending order. As we move down the matrix, the confidences decrease.
- **lower** The lower confidence limit of an edge as estimated by conf.edge.lower.

Author(s)

Michail Tsagris

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

References


Variable selection using the PC-simple algorithm

See Also

`pc.skel, mmhc.skel, corfs.network, local.mmhc.skel, conf.edge.lower`

Examples

## simulate a dataset with continuous data
```r
y <- rdag2(500, p = 25, nei = 3)
ind <- sample(1:25, 25)
x <- y$x[, ind]
mod <- pc.skel( x, method = "comb.fast", alpha = 0.01 )
G <- y$G[ind, ind]
G <- G + t(G)
bn.skel.utils(mod, G, roc = FALSE, alpha = 0.01)
bn.skel.utils2(mod, G, roc = FALSE, alpha = 0.01)
```

---

**Description**

Variable selection using the PC-simple algorithm.

**Usage**

```r
pc.sel(target, dataset, threshold = 0.05)
```

**Arguments**

- `target`: A numerical vector with continuous data.
- `dataset`: A matrix with numerical data; the independent variables, of which some will probably be selected.
- `threshold`: The significance level.

**Details**

Variable selection for continuous data only is performed using the PC-simple algorithm (Buhlmann, Kalisch and Maathuis, 2010). The PC algorithm used to infer the skeleton of a Bayesian Network has been adopted in the context of variable selection. In other words, the PC algorithm is used for a single node.

**Value**

A list including:

- `vars`: A vector with the selected variables.
- `n.tests`: The number of tests performed.
- `runtime`: The runtime of the algorithm.
**Zero inflated Poisson and negative binomial regression**

**Author(s)**
Michail Tsagris

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

**References**

**See Also**
pc.skel, omp

**Examples**
```r
y <- rnorm(100)
x <- matrix(rnorm(100 * 30), ncol = 30)
a <- MXM::pc.sel(y, x)
b <- MMPC(y, x)
```

---

**Description**

Zero inflated Poisson and negative binomial regression.

**Usage**
```r
zip.mod(target, dataset, wei = NULL, xnew = NULL)
zip.reg(target, dataset, wei = NULL, lgy = NULL)
zinb.mod(target, dataset, xnew = NULL)
zinb.reg(target, dataset, lgy = NULL)
```

**Arguments**

- **target**: The target (dependent) variable. It must be a numerical vector with integers.
- **dataset**: The independent 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. If this is NULL, a zero inflated Poisson distribution is fitted, no covariates are present.
- **wei**: A vector of weights to be used for weighted regression. The default value is NULL. An example where weights are used is surveys when stratified sampling has occurred. This is applicable only in the zero inflated Poisson distribution.
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. The zero inflated negative binomial regression does not accept weights though.

Value
A list including:

be The estimated coefficients of the model and for the zip.mod and zinb.mod the standard errors, Wald test statistics and p-values are included.

prop The estimated proportion of zeros.

loglik The log-likelihood of the regression model.

est The estimated values if "xnew" is not NULL.

Author(s)
Michail Tsagris
R implementation and documentation: Michail Tsagris <mtsagris@uoc.gr>

References


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

* Ancestors
  Ancestors and descendants of a node in a directed graph, 8
* Area under the curve
  Conditional independence tests counting the number of times a possible collider d-separates two nodes, 63
  Drop all possible single terms from a model using the partial correlation, 118
  Effective sample size for G^2 test in BNs with case control data, 121
  ROC and AUC, 213
  Supervised PCA, 229
* Backward regression
  Backward selection regression, 11
  IAMB backward selection phase, 164
  IAMB variable selection, 166
* Beta regression
  Conditional independence test for circular data, 49
  Conditional independence test for proportions/percentages, 58
* Binary logistic regression
  Conditional independence test for binary, categorical or ordinal data, 43
  Conditional independence test for the static-longitudinal scenario, 60
* Binomial logistic regression
  Conditional independence tests for success rates, 77
* Case-control studies
  Conditional independence test for case control data, 47
* Conditional Independence Test
  Correlation based conditional independence tests, 101
* Conditional independence tests
  Conditional independence test for binary, categorical or ordinal data, 43
  Conditional independence test for case control data, 47
  Conditional independence test for circular data, 49
  Conditional independence test for longitudinal and clustered data using GLMM, 55
  Conditional independence test for proportions/percentages, 58
  Conditional independence test for the static-longitudinal scenario, 60
  Conditional independence tests for continuous univariate and multivariate data, 65
  Conditional independence tests for count data, 68
  Conditional independence tests for left censored data, 71
  Conditional independence tests for success rates, 77
  G-square conditional independence test for discrete data, 151
* Conditional logistic regression
  Conditional independence test for case control data, 47
* Conditioning set
  Partial correlation between two variables, 198
* Cross validation

251
Bootstrap bias correction for the performance of the cross-validation procedure, 26
Cross-Validation for SES and MMPC, 109
* **DAGs**
  - Check Markov equivalence of two DAGs, 30
* **DAG**
  - Topological sort of a DAG, 235
* **Descendants**
  - Ancestors and descendants of a node in a directed graph, 8
* **Directed acyclic graph**
  - Check whether a directed graph is acyclic, 31
* **Directed acyclic graph**
  - Data simulation from a DAG, 116
* **Essential graph**
  - Transformation of a DAG into an essential graph, 238
* **Feature Selection**
  - Backward phase of MMPC, 9
  - Constraint based feature selection algorithms, 85
  - Constraint based feature selection algorithms for multiple datasets, 96
  - Fast MMPC, 123
  - Feature selection using SES and MMPC for classification with longitudinal data, 130
  - MMPC solution paths for many combinations of hyper-parameters, 181
  - MXM-package, 5
* **Fisher’s Test**
  - Correlation based conditional independence tests, 101
* **Fisher’s test**
  - Conditional independence tests with and without permutation p-value, 83
* **G-square test**
  - G-square conditional independence test for discrete data, 151
* **Gamma regression**
  - Conditional independence tests for positive data, 74
* **Generation of folds**
  - Generate random folds for cross-validation, 157
* **Graphical visualisation**
  - Plot of longitudinal data, 200
* **Internal MXM Functions**
  - MXM-internal, 187
* **Linear mixed model**
  - Conditional independence test for longitudinal and clustered data using GLMM, 55
* **Linear regression test**
  - Conditional independence tests for continuous univariate and multivariate data, 65
* **Log likelihood ratio**
  - Conditional independence tests for left censored data, 71
  - G-square conditional independence test for discrete data, 151
* **Log link**
  - Conditional independence tests for positive data, 74
* **Longitudinal data**
  - Constraint based feature selection algorithms for longitudinal and clustered data, 91
  - Fast MMPC for longitudinal and clustered data, 126
  - Plot of longitudinal data, 200
* **MMHC algorithm**
  - Bayesian Network construction using a hybrid of MMPC and PC, 18
  - Skeleton (local) around a node of the max-min hill-climbing (MMHC) algorithm, 218
  - Skeleton of the max-min hill-climbing (MMHC) algorithm, 220
* **MMPC output**
  - MMPCoutput-class, 186
* **MMPC.glmm output**
  - MMPC.glmm.output-class, 185
* **MMPC**
  - Bootstrap bias correction for the performance of the
INDEX

cross-validation procedure, 26
Cross-Validation for SES and MMPC, 109

* Markov Blanket
Backward selection regression, 11
Backward selection with generalised linear regression models, 17
BIC based forward selection, 22
BIC based forward selection with generalised linear models, 24
Forward selection regression, 145
Forward selection with generalised linear regression models, 148
Forward selection with linear regression models, 149
IAMB backward selection phase, 164
IAMB variable selection, 166
The max-min Markov blanket algorithm, 233

* Markov equivalence
Check Markov equivalence of two DAGs, 30

* Mixed models
Constraint based feature selection algorithms for longitudinal and clustered data, 91
Fast MMPC for longitudinal and clustered data, 126

* Multinomial logistic regression
Conditional independence test for binary, categorical or ordinal data, 43
Conditional independence test for the static-longitudinal scenario, 60

* Multiple Feature Signatures
Backward phase of MMPC, 9
Constraint based feature selection algorithms, 85
Constraint based feature selection algorithms for multiple datasets, 96
Fast MMPC, 123
Feature selection using SES and MMPC for classification with longitudinal data, 130
MMPC solution paths for many combinations of hyper-parameters, 181
MXM-package, 5

* Negative binomial regression
Conditional independence tests for count data, 68

* Neighbour nodes
Markov Blanket of a node in a directed graph, 179
Neighbours of nodes in an undirected graph, 193

* Network construction
Network construction using the partial correlation based forward regression or FBED, 194

* Network plot
Interactive plot of an (un)directed graph, 169
Markov Blanket of a node in a directed graph, 179
Neighbours of nodes in an undirected graph, 193

* Ordinal logistic regression
Conditional independence test for binary, categorical or ordinal data, 43
Conditional independence test for the static-longitudinal scenario, 60

* PC algorithm
Orientation rules for the PC algorithm, 196

* Partial correlation
Partial correlation between two variables, 198

* Permutation test
Permutation based p-value for the Pearson correlation coefficient, 199

* Poisson regressions
Many Wald based tests for logistic and Poisson regressions with continuous predictors, 178

* Poisson regression
Conditional independence tests for count data, 68

* Regression modelling
Generalised linear mixed models
Regression models based on glmm SES and MMPC outputs, 153
Regression models based on SES and MMPC outputs, 205
Regression models based on SES.timeclass and MMPC.timeclass outputs, 207

* Regression models
  Conditional independence tests, 32
  Conditional independence regression based tests, 40
  eBIC for many regression models, 119
  Many simple beta regressions, 174
  Many simple zero inflated Poisson regressions, 176
  Univariate regression based tests, 241

* Regression model
  Beta regression, 21
  Generalised ordinal regression, 156
  Regression modeler, 203
  Regression models fitting, 208

* Ridge regression
  Cross-validation for ridge regression, 107
  Ridge regression, 210
  Ridge regression coefficients plot, 212

* SES output
  SESoutput-class, 217

* SES.glmm output
  SES.gee.output-class, 215
  SES.glmm.output-class, 216

* SES
  Backward phase of MMPC, 9
  Bootstrap bias correction for the performance of the cross-validation procedure, 26
  Constraint based feature selection algorithms, 85
  Constraint based feature selection algorithms for longitudinal and clustered data, 91
  Constraint based feature selection algorithms for multiple datasets, 96
  Cross-Validation for SES and MMPC, 109
  Fast MMPC, 123
  Fast MMPC for longitudinal and clustered data, 126
  Feature selection using SES and MMPC for classifiication with longitudinal data, 130
  MMPC solution paths for many combinations of hyper-parameters, 181
  MXM-package, 5

* Structural Hamming distance
  Structural Hamming distance between two partially oriented DAGs, 227

* Survival
  Conditional independence tests for left censored data, 71

* Tobit regression
  Conditional independence tests for left censored data, 71

* Topological sort
  Topological sort of a DAG, 235

* Undirected path
  Undirected path(s) between two nodes, 240

* Variable Selection
  Backward phase of MMPC, 9
  Backward selection regression, 11
  Backward selection with generalised linear regression models, 17
  BIC based forward selection, 22
  BIC based forward selection with generalised linear models, 24
  Constraint based feature selection algorithms, 85
  Constraint based feature selection algorithms for longitudinal and clustered data, 91
  Constraint based feature selection algorithms for multiple datasets, 96
  Fast MMPC, 123
  Fast MMPC for longitudinal and clustered data, 126
  Feature selection using SES and
MMPC for classification with longitudinal data, 130
Forward selection regression, 145
Forward selection with generalised linear regression models, 148
Forward selection with linear regression models, 149
IAMB backward selection phase, 164
IAMB variable selection, 166
MMPC solution paths for many combinations of hyper-parameters, 181
The max-min Markov blanket algorithm, 233
* Wald test
  Many Wald based tests for logistic and Poisson regressions with continuous predictors, 178
* Zero inflated poisson regression
  Conditional independence tests for count data, 68
* beta distribution
  Beta regression, 21
  Many simple beta regressions, 174
* check for cycles
  Check whether a directed graph is acyclic, 31
* conditional independence test
  Conditional independence tests with and without permutation p-value, 83
  Permutation based p-value for the Pearson correlation coefficient, 199
* directed graph
  Interactive plot of an (un)directed graph, 169
* distance between DAGs
  Structural Hamming distance between two partially oriented DAGs, 227
* equivalence class
  Transformation of a DAG into an essential graph, 238
* forward regression
  Network construction using the partial correlation based forward regression or FBED, 194
* interactive graph
  Interactive plot of an (un)directed graph, 169
* logistic regressions
  Many Wald based tests for logistic and Poisson regressions with continuous predictors, 178
* meta analytic ses output
  mases.output-class, 180
* meta-analytic MMPC output
  mammpc.output-class, 171
* ordinal regression
  Generalised ordinal regression, 156
* parallel
  Bootstrap bias correction for the performance of the cross-validation procedure, 26
  Cross-Validation for SES and MMPC, 109
* pc algorithm
  Skeleton of the PC algorithm, 223
* permutation based p-value
  Conditional independence tests with and without permutation p-value, 83
* receiver operating curve
  Conditional independence tests counting the number of times a possible collider d-separates two nodes, 63
  Drop all possible single terms from a model using the partial correlation, 118
  Effective sample size for G^2 test in BNs with case control data, 121
  ROC and AUC, 213
  Supervised PCA, 229
* simulation of random values
  Data simulation from a DAG, 116
* zero inflated Poisson
  Many simple zero inflated Poisson regressions, 176
acc.mxm (Cross-Validation for SES and MMPC), 109
acc_multinom.mxm (Cross-Validation for SES and MMPC), 109
Ancestors and descendants of a node in a directed graph, 8
anova, 73, 82
apply_ideq (MXM-internal), 187
auc, 246
auc (ROC and AUC), 213
auc.mxm (Cross-Validation for SES and MMPC), 109
Backward phase of MMPC, 9
Backward selection regression, 11
Backward selection regression for GLMM, 12
Backward selection regression for GLMM using the eBIC, 14
Backward selection regression using the eBIC, 15
Backward selection with generalised linear regression models, 17
Bayesian Network construction using a hybrid of MMPC and PC, 18
bbc, 105, 106, 110, 111, 113, 214
bbc (Bootstrap bias correction for the performance of the cross-validation procedure), 26
Beta regression, 21
beta.bsreg (MXM-internal), 187
beta.fsreg (MXM-internal), 187
beta.fsreg_2 (MXM-internal), 187
beta.mod (Beta regression), 21
beta.mxm (Cross-Validation for SES and MMPC), 109
beta.reg (MXM-internal), 187
beta.regs, 22
beta.regs (Many simple beta regressions), 174
betamle.wei (MXM-internal), 187
BIC based forward selection, 22
BIC based forward selection with generalised linear models, 24
bic.betafsreg (MXM-internal), 187
bic.clogit.fsreg (MXM-internal), 187
bic.fsreg, 12, 17, 18, 25, 136, 139, 147, 149, 151, 166, 167
bic.fsreg (BIC based forward selection), 22
bic.gammafsreg (BIC based forward selection with generalised linear models), 24
bic.glm.fsreg, 12, 18, 24, 147, 149, 151, 166, 167
bic.glm.fsreg (BIC based forward selection with generalised linear models), 24
bic.llr.fsreg (MXM-internal), 187
bic.mm.fsreg (BIC based forward selection with generalised linear models), 24
bic.normlog.fsreg (BIC based forward selection with generalised linear models), 24
bic.regs, 173, 176
bic.tobit.fsreg (MXM-internal), 187
bic.wf.fsreg (MXM-internal), 187
bic.zipfsreg (MXM-internal), 187
big.bs.g2 (MXM-internal), 187
big.fbed.g2 (MXM-internal), 187
big.fbed.reg (Forward Backward Early Dropping selection regression for big data), 137
big.gomp, 138, 203
big.gomp (Generic orthogonal matching pursuit (gOMP) for big data), 161
big.model (MXM-internal), 187
big.score.univregs (Univariate regression based tests), 241
bn.skel.utils, 123, 164, 222, 223, 225, 227
bn.skel.utils (Utilities for the skeleton of a (Bayesian) Network), 245
bn.skel.utils2, 170
bn.skel.utils2 (Utilities for the skeleton of a (Bayesian) Network), 245
bootstrap/gomp (Generic orthogonal matching pursuit (gOMP)), 158
Bootstrap bias correction for the performance of the cross-validation procedure, 26
bs.g2 (MXM-internal), 187
bs.reg (Backward selection regression), 11
bsreg.big (MXM-internal), 187
Calculation of the constant and slope for each subject over time, 28
INDEX

cat.ci, 232, 233
cat.ci (Conditional independence tests with and without permutation p-value), 83
cat_condis (MXM-internal), 187
censIndCR, 8, 49, 113, 153
censIndCR (Conditional independence tests for survival data), 79
censIndER (Conditional independence tests for survival data), 79
censIndLLR (Conditional independence tests for survival data), 79
censIndWR, 49, 73, 82
censIndWR (Conditional independence tests for survival data), 79
Certificate of exclusion from the selected variables set using SES or MMPC, 29
certificate.of.exclusion (Certificate of exclusion from the selected variables set using SES or MMPC), 29
certificate.of.exclusion2, 125
certificate.of.exclusion2 (Certificate of exclusion from the selected variables set using SES or MMPC), 29
Check Markov equivalence of two DAGs, 30
Check whether a directed graph is acyclic, 31
ci.fast (Symmetric conditional independence test with mixed data), 232
ci.fast2 (Symmetric conditional independence test with mixed data), 232
ci.mm, 118
ci.mm (Symmetric conditional independence test with mixed data), 232
ci.mm2 (Symmetric conditional independence test with mixed data), 232
ci.mm (Cross-Validation for SES and MMPC), 109
ciwr.mm (Cross-Validation for SES and MMPC), 109
clogit.fsreg (MXM-internal), 187
clogit.fsreg_2 (MXM-internal), 187
clogit.mm (Cross-Validation for SES and MMPC), 109
comb_condis (MXM-internal), 187
compare_p_values (MXM-internal), 187
cond.reg, 245
cond.reg (Conditional independence regression based tests), 40
condi, 231–233
condi (Conditional independence tests with and without permutation p-value), 83
condi.perm (MXM-internal), 187
CondIndTests, 9–12, 18, 22, 24, 25, 43, 46, 51, 54, 57, 60, 62, 68, 71, 73, 76, 79, 82, 84, 86, 87, 90, 96–98, 100, 104, 113, 120, 129, 147, 149, 151, 153, 166, 167, 182, 183, 200, 204, 210, 235, 245
CondIndTests (Conditional independence tests), 32
CondIndTests, 9–12, 18, 22, 24, 25, 43, 46, 51, 54, 57, 60, 62, 68, 71, 73, 76, 79, 82, 84, 86, 87, 90, 96–98, 100, 104, 113, 120, 129, 147, 149, 151, 153, 166, 167, 182, 183, 200, 204, 210, 235, 245
CondIndTests (Conditional independence tests), 32
condi (Conditional independence tests counting the number of times a possible collider d-separates two nodes), 63
Conditional independence regression based tests, 40
Conditional independence test for binary, categorical or ordinal data, 43
Conditional independence test for case control data, 47
Conditional independence test for circular data, 49
Conditional independence test for longitudinal and clustered data using GEE, 51
Conditional independence test for longitudinal and clustered data using GLMM, 55
Conditional independence test for proportions/percentages, 58
Conditional independence test for the static-longitudinal scenario, 60
Conditional independence tests
counting the number of times a possible collider d-separates two nodes, 63

Conditional independence tests for continuous univariate and multivariate data, 65

Conditional independence tests for count data, 68

Conditional independence tests for left censored data, 71

Conditional independence tests for positive data, 74

Conditional independence tests for success rates, 77

Conditional independence tests for survival data, 79

Conditional independence tests with and without permutation p-value, 83

conf.edge.lower, 123, 134, 246, 247

conf.edge.lower (Lower limit of the confidence of an edge), 170

Constraint based feature selection algorithms, 85

Constraint based feature selection algorithms for longitudinal and clustered data, 91

Constraint based feature selection algorithms for multiple datasets, 96

cor.drop1 (Drop all possible single terms from a model using the partial correlation), 118

cor.fbed, 160

cor.fsreg, 160

corfbed.network (Network construction using the partial correlation based forward regression or FBED), 194

corfs.network, 20, 123, 134, 164, 219, 223, 227, 247

Correlation based conditional independence tests, 101
correls, 160
coxph.mxm (Cross-Validation for SES and MMPC), 109

Cross-Validation for gOMP, 104

Cross-validation for ridge regression, 107

Cross-Validation for SES and MMPC, 109

Cross-validation of the FBED with LMM, 114

cv.fbed.lmm.reg, 145
cv.fbed.lmm.reg (Cross-validation of the FBED with LMM), 114

cv.gomp, 27, 113

cv.gomp (Cross-Validation for gOMP), 104
cv.mmmpc, 10, 106, 155, 169, 206
cv.mmmpc (Cross-Validation for SES and MMPC), 109
cv.permmmmpc (Cross-Validation for SES and MMPC), 109
cv.permsses (Cross-Validation for SES and MMPC), 109
cv.ses, 26, 27, 90, 100, 106, 155, 158, 169, 183, 206
cv.ses (Cross-Validation for SES and MMPC), 109
cv.waldmmmpc (Cross-Validation for SES and MMPC), 109
cv.waldsses (Cross-Validation for SES and MMPC), 109
cvlogit.cv.ses (MXM-internal), 187
cvmmpc.par (MXM-internal), 187
cvpermmmpc.par (MXM-internal), 187
cvpermses.par (MXM-internal), 187
cvses.par (MXM-internal), 187
cwalddmmmpc.par (MXM-internal), 187
cwalddses.par (MXM-internal), 187
cwlegacy.32
dag2eg (Transformation of a DAG into an essential graph), 238
dag_to_eg (MXM-internal), 187

Data simulation from a DAG, 116
disctor_condis (MXM-internal), 187
dist_condi, 232
dist_condi (Conditional independence tests with and without permutation p-value), 83
Drop all possible single terms from a model using the partial correlation, 118
drop1, 118

eBIC for many regression models, 119
ebic.beta.bsreg (MXM-internal), 187
ebic.bsreg, 136, 139
ebic.bsreg (Backward selection regression using the eBIC), 15
ebic.cr.bsreg (MXM-internal), 187
ebic.fbed.beta (MXM-internal), 187
ebic.fbed.cr (MXM-internal), 187
ebic.fbed(glm) (MXM-internal), 187
ebic.fbed.glmm (MXM-internal), 187
ebic.fbed.lm (MXM-internal), 187
ebic.fbed.1mm (MXM-internal), 187
ebic.fbed.mmmreg (MXM-internal), 187
ebic.fbed.multinom (MXM-internal), 187
ebic.fbed.nb (MXM-internal), 187
ebic.fbed.ordinal (MXM-internal), 187
ebic.fbed.tobit (MXM-internal), 187
ebic.fbed.wr (MXM-internal), 187
ebic.fbed.zip (MXM-internal), 187
ebic.glmm.bsreg (MXM-internal), 187
ebic.glmm.mbsreg, 13
ebic.glmm.bsreg (Backward selection regression for GLMM using the eBIC), 14
ebic.glmm.cr.bsreg (MXM-internal), 187
ebic.glmm.ordinals.reps.bsreg (MXM-internal), 187
ebic.glmm.reps.bsreg (MXM-internal), 187
ebic.lr.bsreg (MXM-internal), 187
ebic.ls.bsreg (MXM-internal), 187
ebic.mm.bsreg (MXM-internal), 187
ebic.model (MXM-internal), 187
ebic.multinom.bsreg (MXM-internal), 187
ebic.ordinals.bsreg (MXM-internal), 187
ebic.regs (eBIC for many regression models), 119
ebic.spml.bsreg (MXM-internal), 187
ebic.tobit.bsreg (MXM-internal), 187
ebic.univregs (Univariate regression based tests), 241
ebic.wr.bsreg (MXM-internal), 187
ebic.zip.bsreg (MXM-internal), 187
ebicScore (MXM-internal), 187

Effective sample size for G^2 test in BNs with case control data, 121
equivdags (Check Markov equivalence of two DAGs), 30
Estimation of the percentage of Null p-values, 122
euclid_sens.spec.mxm (Cross-Validation for SES and MMPC), 109
exporeg.mxm (Cross-Validation for SES and MMPC), 109

Fast MMPC, 123
Fast MMPC for longitudinal and clustered data, 126
fbed.ebic (MXM-internal), 187
fbed.g2 (MXM-internal), 187
fbed.gee.reg, 29, 115, 145
fbed.gee.reg (Forward Backward Early Dropping selection regression with GEE), 139
fbed.geelm (MXM-internal), 187
fbed.gee.glm (MXM-internal), 187
fbed.glmm (MXM-internal), 187
fbed.glmm.reg, 13, 15, 115, 142
fbed.glmm.reg (Forward Backward Early Dropping selection regression with GLMM), 142
fbed.1mm (MXM-internal), 187
fbed.lr (MXM-internal), 187
fbed.ordgee (MXM-internal), 187
fbed.reg, 8, 119, 142, 168, 230
fbed.reg (Forward Backward Early Dropping selection regression), 134
fbedreg.bic, 204, 210
fbedreg.bic (Incremental BIC values and final regression model of the FBED algorithm), 168
Feature selection using SES and MMPC for classification with longitudinal data, 130
findAncestors (Ancestors and descendants of a node in a directed graph), 8
findDescendants (Ancestors and descendants of a node in a directed graph), 8
Fit a mixture of beta distributions in p-values, 133
Forward Backward Early Dropping selection regression, 134
Forward Backward Early Dropping selection regression for big data, 137
Forward Backward Early Dropping selection regression with GEE, 139
Forward Backward Early Dropping selection regression with GLMM, 142
Forward selection regression, 145
Forward selection with generalised linear regression models, 148
Forward selection with linear regression models, 149
fs.reg, 17, 18, 25, 136, 139, 149, 151, 160
fs.reg (Forward selection regression), 145
fs.reg_2 (MXM-internal), 187
fscore.mx (Cross-Validation for SES and MMPC), 109
G-square conditional independence test for discrete data, 151
gammafsreg (Forward selection with generalised linear regression models), 148
gammafsreg_2 (MXM-internal), 187
gee.ci.mm (Symmetric conditional independence test with clustered data), 230
ggee.condregs (Conditional independence regression based tests), 40
ggee.mmhc.skel (Skeleton of the max-min hill-climbing (MMHC) algorithm), 220
glmm.nb.bsreg (MXM-internal), 187
glmm.nb.reps.bsreg (MXM-internal), 187
glmm.ordinal.bsreg (MXM-internal), 187
glmm.ordinal.reps.bsreg (MXM-internal), 187
glmm.pc.skel (Skeleton of the PC algorithm), 223
glmm.univregs (Univariate regression based tests), 241
Generalised linear mixed models based on glmm SES and MMPC outputs, 153
Generalised ordinal regression, 156
Generate random folds for cross-validation, 157
generatefolds (Generate random folds for cross-validation), 157
Generic orthogonal matching pursuit (gOMP), 158
Generic orthogonal matching pursuit (gOMP) for big data, 161
glm, 178
glm.bsreg, 119
glm.bsreg (Backward selection with generalised linear regression models), 17
glm.bsreg_2 (Backward selection with generalised linear regression models), 17
glm.fsreg, 12, 24, 147, 166, 167
glm.fsreg (Forward selection with generalised linear regression models), 148
gglm.fsreg_2 (MXM-internal), 187
glm.mx (Cross-Validation for SES and MMPC), 109
glms.bsreg, 15, 29, 142, 145
glms.bsreg (Backward selection regression for GLMM), 12
glmm.ci.mm (Symmetric conditional independence test with clustered data), 230
glmm.condregs (Conditional independence regression based tests), 40
glmm.cr.bsreg (MXM-internal), 187
glmm.mmhc.skel (Skeleton of the max-min hill-climbing (MMHC) algorithm), 220
glmm.nb.bsreg (MXM-internal), 187
glmm.nb.reps.bsreg (MXM-internal), 187
glmm.ordinal.bsreg (MXM-internal), 187
glmm.ordinal.reps.bsreg (MXM-internal), 187
glmm.pc.skel (Skeleton of the PC algorithm), 223
glmm.univregs (Univariate regression based tests), 241
gomp, 8, 163, 230
gomp (Generic orthogonal matching pursuit (gOMP)), 158
gomp.path, 106
gomp2 (MXM-internal), 187
Graph of unconditional associations, 163
INDEX

group.mvbetas, 61
group.mvbetas (Calculation of the constant and slope for each subject over time), 28
gSquare, 8, 46, 62, 71, 73, 76, 82, 84, 104, 113
gSquare (G-square conditional independence test for discrete data), 151

iamb (IAMB variable selection), 166
IAMB backward selection phase, 164
IAMB variable selection, 166
iamb.betabs (MXM-internal), 187
iamb.bs (IAMB backward selection phase), 164
iamb.gammabs (MXM-internal), 187
iamb.glmb (MXM-internal), 187
iamb.normlogibs (MXM-internal), 187
iamb.tobitbs (MXM-internal), 187
iamb.zipbs (MXM-internal), 187
ida (Total causal effect of a node on another node), 236
IdentifyEquivalence (MXM-internal), 187
identifyTheEquivalent (MXM-internal), 187
Incremental BIC values and final regression model of the FBED algorithm, 168
Interactive plot of an (un)directed graph, 169
internaliamb.betabs (MXM-internal), 187
internaliamb.binombs (MXM-internal), 187
internaliamb.gammabs (MXM-internal), 187
internaliamb.lmbs (MXM-internal), 187
internaliamb.mmbs (MXM-internal), 187
internaliamb.normlogibs (MXM-internal), 187
internaliamb.poisibs (MXM-internal), 187
internaliamb.tobitibs (MXM-internal), 187
internaliamb.zipibs (MXM-internal), 187
is.dag, 197, 238
is.dag (Check whether a directed graph is acyclic), 31
is.sepset (MXM-internal), 187

kfbed.gee.reg (MXM-internal), 187
kfbed.glmm.reg (MXM-internal), 187
kfbed.reg (MXM-internal), 187
llr.bsreg (MXM-internal), 187
llrreg.mxm (Cross-Validation for SES and MMPC), 109
lm.fsreg, 12, 18, 24, 25, 147, 149, 151, 166, 167
lm.fsreg (Forward selection with linear regression models), 149
lm.fsreg_2 (MXM-internal), 187
lm.mm (Cross-Validation for SES and MMPC), 109
lm.mmbsreg (MXM-internal), 187
lmrob.mxm (Cross-Validation for SES and MMPC), 109
local.mmhc.skel, 123, 134, 227, 247
local.mmhc.skel (Skeleton (local) around a node of the max-min hill-climbing (MMHC) algorithm), 218
logiquant.regs, 173
logiquant.regs (Many simple quantile regressions using logistic regressions), 175
Lower limit of the confidence of an edge, 170

ma.mmpc, 172, 181
ma.mmpc (Constraint based feature selection algorithms for multiple datasets), 96
ma.ses, 172, 181
ma.ses (Constraint based feature selection algorithms for multiple datasets), 96
mae.mxm (Cross-Validation for SES and MMPC), 109
mammpc.output (mammpc.output-class), 171
mammpc.output-class, 171
mammpc.output-method (mammpc.output-class), 171
Many approximate simple logistic regressions, 172
Many simple beta regressions, 174
Many simple quantile regressions using logistic regressions, 175
Many simple zero inflated Poisson regressions, 176
Many Wald based tests for logistic and Poisson regressions with continuous predictors, 178
Markov Blanket of a node in a directed graph, 179
mases.output (mases.output-class), 180
mases.output-class, 180
mases.output-method (mases.output-class), 180
max_min_assoc (MXM-internal), 187
mb, 9, 170, 197
mb (Markov Blanket of a node in a directed graph), 179
mci.mxm (Cross-Validation for SES and MMPC), 109
min_assoc (MXM-internal), 187
mmhc.skel (Skeleton of the max-min hill-climbing (MMHC) algorithm), 220
mmmb (The max-min Markov blanket algorithm), 233
MMPC, 8, 10, 12, 17, 18, 24, 25, 28, 30, 43, 64, 120, 122, 124, 125, 136, 139, 147, 149, 151, 155, 166, 167, 169, 186, 193, 206, 220, 230, 235, 241, 245
MMPC (Constraint based feature selection algorithms), 85
MMPC solution paths for many combinations of hyper-parameters, 181
MMPC.gee, 184, 216
MMPC.gee (Constraint based feature selection algorithms for longitudinal and clustered data), 91
mmpc.ggee.model (Generalised linear mixed models based on glmm SES and MMPC outputs), 153
MMPC.ggee.output (MMPC.ggee.output-class), 184
MMPC.ggee.output-class, 184
MMPC.ggee.output-method (MMPC.ggee.output-class), 184
mmpc.ggee2 (Fast MMPC for longitudinal and clustered data), 126
MMPC.glmm, 13, 15, 29, 54, 57, 113, 142, 145, 185, 217, 220
MMPC.glmm (Constraint based feature selection algorithms for longitudinal and clustered data), 91
mmpc.glmm.model (Generalised linear mixed models based on glmm SES and MMPC outputs), 153
MMPC.glmm.output (MMPC.glmm.output-class), 185
MMPC.glmm.output-class, 185
MMPC.glmm.output-method (MMPC.glmm.output-class), 185
mmpc.glmm2 (Fast MMPC for longitudinal and clustered data), 126
mmpc.model, 169, 204, 210
mmpc.model (Regression models based on SES and MMPC outputs), 205
mmpc.or (Bayesian Network construction using a hybrid of MMPC and PC), 18
mmpc.path (MMPC solution paths for many combinations of hyper-parameters), 181
MMPC.timeclass, 208
MMPC.timeclass (Feature selection using SES and MMPC for classification with longitudinal data), 130
mmpc.timeclass.model, 133
mmpc.timeclass.model (Regression models based on SES.timeclass and MMPC.timeclass outputs), 207
mmpc2, 128
mmpc2 (Fast MMPC), 123
mmpcbackphase, 119, 221
mmpcbackphase (Backward phase of MMPC), 9
MMPCoutput (MMPCoutput-class), 186
MMPCoutput-class, 186
MMPCoutput-method (MMPCoutput-class), 186
modeler, 210
modeler (Regression modeler), 203
mse.mxm (Cross-Validation for SES and MMPC), 109
multinom.mxm (Cross-Validation for SES and MMPC), 109
MXM-internal, 187
MXM-package, 5
nb.mxm (Cross-Validation for SES and MMPC), 109
nbdev.mxm (Cross-Validation for SES and MMPC), 109
nchoosek (MXM-internal), 187
nchoosek (MXM-internal), 187
nei, 9, 170, 180, 236, 237
nei (Neighbours of nodes in an undirected graph), 193
Neighbours of nodes in an undirected graph, 193
Ness (Effective sample size for G^2 test in BNs with case control data), 121
Network construction using the partial correlation based forward regression or FBED, 194
normlog.fsreg (Forward selection with generalised linear regression models), 148
omp, 248
ord.resid (Probability residual of ordinal logistic regression), 201
ord_mae.mxm (Cross-Validation for SES and MMPC), 109
ordinal.mxm (Cross-Validation for SES and MMPC), 109
ordinal.reg, 202
ordinal.reg (Generalised ordinal regression), 156
Orientation rules for the PC algorithm, 196
par, 200
Partial correlation between two variables, 198
partialcor (Partial correlation between two variables), 198
pc.con, 31, 169, 196–198
pc.con (Skeleton of the PC algorithm), 223
pc.or, 8, 9, 20, 30–32, 118, 179, 180, 219, 223, 228, 236–238
pc.or (Orientation rules for the PC algorithm), 196
pc.sel, 8
pc.sel (Variable selection using the PC-simple algorithm), 247
pc.skel (Skeleton of the PC algorithm), 223
pc.skel.boot, 171
pearson_condis (MXM-internal), 187
perm.apply_ideq (MXM-internal), 187
perm.betaregs (Many simple beta regressions), 174
perm.IdentifyEquivalence (MXM-internal), 187
perm.identifyTheEquivalent (MXM-internal), 187
perm.mmpc (Constraint based feature selection algorithms), 85
perm.mmpc.path (MMPC solution paths for many combinations of hyper-parameters), 181
perm.ses (Constraint based feature selection algorithms), 85
perm.univariateScore (MXM-internal), 187
perm.univregs, 179
perm.univregs (Univariate regression based tests), 241
perm.zipregs (Many simple zero inflated Poisson regressions), 176
permBeta (Conditional independence test for proportions/percentages), 58
permBinom (Conditional independence tests for success rates), 77
permClogit (Conditional independence test for case control data), 47
permcor, 198
permcor (Permutation based p-value for the Pearson correlation coefficient), 199
permcorrels (Permutation based p-value for the Pearson correlation coefficient), 199
permCR (Conditional independence tests for survival data), 79
permDcor (Correlation based conditionional independence tests), 101
permER (Conditional independence tests for survival data), 79
permFisher (Correlation based conditionional independence tests), 101
permGamma (Conditional independence tests for positive data), 74
permSquare (G-square conditionional independence test for discrete data), 151
permIGreg (Conditional independence tests for positive data), 74
permLLR (Conditional independence tests for survival data), 79
permLogistic (Conditional independence test for binary, categorical or ordinal data), 43
permMMFisher (Correlation based conditionional independence tests), 101
permMMReg (Conditional independence tests for continuous univariate and multivariate data), 65
permMultinom (Conditional independence test for binary, categorical or ordinal data), 43
permMVreg (Conditional independence tests for continuous univariate and multivariate data), 65
permNB (Conditional independence tests for count data), 68
permNormLog (Conditional independence tests for positive data), 74
permOrdinal (Conditional independence test for binary, categorical or ordinal data), 43
permPois (Conditional independence tests for count data), 68
permReg (Conditional independence tests for continuous univariate and multivariate data), 65
permRQ (Conditional independence tests for continuous univariate and multivariate data), 65
permTobit (Conditional independence tests for left censored data), 71
Permutation based p-value for the Pearson correlation coefficient, 199
permWR (Conditional independence tests for survival data), 79
permZIP (Conditional independence tests for count data), 68
pi0est (Estimation of the percentage of Null p-values), 122
plot, 200
Plot of longitudinal data, 200
plot, mammppc.output, ANY-method
(plot, mammppc.output-class), 171
plot, mammppc.output-method
(plot, mammppc.output-class), 171
plot, mases.output, ANY-method
(plot, mases.output-class), 180
plot, mases.output-method
(plot, mases.output-class), 180
plot, MMPC.gee.output, ANY-method
(plot, MMPC.gee.output-class), 184
plot, MMPC.gee.output-method
(plot, MMPC.gee.output-class), 184
plot, MMPC.glmm.output, ANY-method
(plot, MMPC.glmm.output-class), 185
plot, MMPC.glmm.output-method
(plot, MMPC.glmm.output-class), 185
plot, MMPCoutput, ANY-method
(plot, MMPCoutput-class), 186
plot, MMPCoutput-method
(plot, MMPCoutput-class), 186
plot, SES.gee.output, ANY-method
(plot, SES.gee.output-class), 215
plot, SES.gee.output-method
(plot, SES.gee.output-class), 215
plot, SES.glmm.output, ANY-method
(plot, SES.glmm.output-class), 216
plot, SES.glmm.output-method
(plot, SES.glmm.output-class), 216
plot, SESoutput, ANY-method
(plot, SESoutput-class), 217
plot, SESoutput-method
(plot, SESoutput-class), 217
plotnetwork, 9, 180, 194, 196, 215, 221, 224,
plotnetwork(Interactive plot of an (un)directed graph), 169
pois.mxm (Cross-Validation for SES and MMPC), 109
poisdev.mxm (Cross-Validation for SES and MMPC), 109
prec.mxm (Cross-Validation for SES and MMPC), 109
Probability residual of ordinal logistic regression, 201
proc_time-class (MXM-internal), 187
pval.mixbeta (fit a mixture of beta distributions in p-values), 133
pve.mxm (Cross-Validation for SES and MMPC), 109
quasibinom.fsreg (MXM-internal), 187
quasibinom.fsreg_2 (MXM-internal), 187
quasipois.fsreg (MXM-internal), 187
quasipois.fsreg_2 (MXM-internal), 187
R0 (MXM-internal), 187
R1 (MXM-internal), 187
R2 (MXM-internal), 187
R3 (MXM-internal), 187
rdag (Data simulation from a DAG), 116
rdag2 (Data simulation from a DAG), 116
Read big data or a big.matrix object, 202
read.big.data, 163
read.big.data (Read big data or a big.matrix object), 202
reg.fit, 22, 43, 120, 169, 175, 177, 204, 245, 249
reg.fit (Regression models fitting), 208
regbeta (MXM-internal), 187
regbetae (MXM-internal), 187
Regression modeler, 203
Regression models based on SES and MMPC outputs, 205
Regression models based on SES.timeclass and MMPC.timeclass outputs, 207
Regression models fitting, 208
regzinb (MXM-internal), 187
regzip (MXM-internal), 187
regzipwei (MXM-internal), 187
Ridge regression, 210
Ridge regression coefficients plot, 212
ridge.plot (Ridge regression coefficients plot), 212
ridge.reg, 22, 43, 108, 120, 157, 175, 177, 204, 210, 212, 245, 249
ridge.reg (Ridge regression), 210
ridgereg.cv (Cross-Validation for ridge regression), 107
rint.regs (Univariate regression based tests), 241
score.glms, 178
score.univregs (Univariate regression based tests), 241
Search for triangles in an undirected graph, 214
sens.mxm (Cross-Validation for SES and MMPC), 109
SES, 8, 11, 12, 18, 22, 24, 25, 28, 43, 46, 49, 51, 60, 62, 64, 73, 82, 100, 104, 113, 120, 122, 147, 149, 151, 153, 155, 166, 167, 169, 186, 193, 200, 206, 218, 235, 241, 245
SES (Constraint based feature selection algorithms), 85
SES.ggee, 184, 216
SES.ggee (Constraint based feature selection algorithms for longitudinal and clustered data), 91
ses.ggee.model (Generalised linear mixed models based on glmm SES and MMPC outputs), 153
SES.ggee.output (SES.ggee.output-class), 215
SES.ggee.output-class, 215
SES.ggee.output-method (SES.ggee.output-class), 215
SES.glmm, 54, 57, 185, 201, 217
SES.glmm (Constraint based feature selection algorithms for longitudinal and clustered data), 91
<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
<th>Page(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ses.glmm.model</td>
<td>Generalised linear mixed models based on glmm SES and MMPC outputs</td>
<td>153</td>
</tr>
<tr>
<td>SES.glmm.output</td>
<td>(SES.glmm.output-class)</td>
<td>216</td>
</tr>
<tr>
<td>SES.glmm.output-class</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SES.glmm.output-method</td>
<td>(SES.glmm.output-class)</td>
<td>216</td>
</tr>
<tr>
<td>ses.model</td>
<td>Regression models based on SES and MMPC outputs</td>
<td>205</td>
</tr>
<tr>
<td>SES.timeclass</td>
<td>(Feature selection using SES and MMPC for classifiication with longitudinal data)</td>
<td>130</td>
</tr>
<tr>
<td>ses.timeclass.model</td>
<td>Regression models based on SES.timeclass and MMPC.timeclass outputs</td>
<td>207</td>
</tr>
<tr>
<td>SESOutput</td>
<td>(SESOutput-class)</td>
<td>217</td>
</tr>
<tr>
<td>SESOutput-class</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SESOutput-method</td>
<td>(SESOutput-class)</td>
<td>217</td>
</tr>
<tr>
<td>shd</td>
<td>Structural Hamming distance between two partially oriented DAGs</td>
<td>227</td>
</tr>
<tr>
<td>Skeleton (local)</td>
<td>around a node of the max-min hill-climbing (MMHC) algorithm</td>
<td>218</td>
</tr>
<tr>
<td>Skeleton of the max-min hill-climbing (MMHC) algorithm</td>
<td></td>
<td>220</td>
</tr>
<tr>
<td>Skeleton of the PC algorithm</td>
<td></td>
<td>223</td>
</tr>
<tr>
<td>sp.logiregs</td>
<td>(Many approximate simple logistic regressions)</td>
<td>172</td>
</tr>
<tr>
<td>spec.mm.x</td>
<td>(Cross-Validation for SES and MMPC)</td>
<td>109</td>
</tr>
<tr>
<td>spml.bsreg</td>
<td>(MXM-internal)</td>
<td>187</td>
</tr>
<tr>
<td>Structural Hamming distance between two partially oriented DAGs</td>
<td></td>
<td>227</td>
</tr>
<tr>
<td>Supervised PCA</td>
<td></td>
<td>229</td>
</tr>
<tr>
<td>supervised.pca</td>
<td>(Supervised PCA)</td>
<td>229</td>
</tr>
<tr>
<td>Surv.</td>
<td></td>
<td>72, 73, 80, 82</td>
</tr>
<tr>
<td>survreg</td>
<td></td>
<td>72</td>
</tr>
<tr>
<td>Symmetric conditional independence test with clustered data</td>
<td></td>
<td>230</td>
</tr>
<tr>
<td>Symmetric conditional independence test with mixed data</td>
<td></td>
<td>232</td>
</tr>
<tr>
<td>tc.plot</td>
<td>(Plot of longitudinal data)</td>
<td>200</td>
</tr>
<tr>
<td>test.maker (MXM-internal)</td>
<td></td>
<td>187</td>
</tr>
<tr>
<td>testIndBeta</td>
<td>(Conditional independence test for proportions/percentages)</td>
<td>58</td>
</tr>
<tr>
<td>testIndBeta (Conditional independence test for proportions/percentages)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>testIndBinom</td>
<td>(Conditional independence tests for success rates)</td>
<td>77</td>
</tr>
<tr>
<td>testIndClogit</td>
<td>(Conditional independence test for case control data)</td>
<td>47</td>
</tr>
<tr>
<td>testIndFisher</td>
<td></td>
<td>8, 60, 68, 73, 82, 84, 87, 98, 113, 153, 198, 200</td>
</tr>
<tr>
<td>testIndFisher (Correlation based conditional independence tests)</td>
<td></td>
<td>101</td>
</tr>
<tr>
<td>testIndGamma</td>
<td>(Conditional independence tests for positive data)</td>
<td>74</td>
</tr>
<tr>
<td>testIndGEGamma</td>
<td></td>
<td>127</td>
</tr>
<tr>
<td>testIndGEGamma (Conditional independence test for longitudinal and clustered data using GEE)</td>
<td>51</td>
<td></td>
</tr>
<tr>
<td>testIndGEELogistic</td>
<td></td>
<td>127</td>
</tr>
<tr>
<td>testIndGEELogistic (Conditional independence test for longitudinal and clustered data using GEE)</td>
<td>51</td>
<td></td>
</tr>
<tr>
<td>testIndGEENormLog</td>
<td></td>
<td>127</td>
</tr>
<tr>
<td>testIndGEENormLog (Conditional independence test for longitudinal and clustered data using GEE)</td>
<td>51</td>
<td></td>
</tr>
<tr>
<td>testIndGEEPois</td>
<td></td>
<td>127</td>
</tr>
<tr>
<td>testIndGEEPois (Conditional independence test for longitudinal and clustered data using GEE)</td>
<td>51</td>
<td></td>
</tr>
<tr>
<td>testIndGEEReg</td>
<td>(Conditional independence test for longitudinal and clustered data using GEE)</td>
<td>51</td>
</tr>
<tr>
<td>testIndGLMMCR</td>
<td>(Conditional independence test for longitudinal and clustered data using GLMM)</td>
<td>55</td>
</tr>
<tr>
<td>testIndGLMMGamma</td>
<td>(Conditional independence test for longitudinal and clustered data using GLMM)</td>
<td>127</td>
</tr>
<tr>
<td>testIndGLMMGamma (Conditional independence test for longitudinal and clustered data using GLMM)</td>
<td>51</td>
<td></td>
</tr>
</tbody>
</table>
data using GLMM, 55
testIndGLMMLogistic, 127, 155
testIndGLMMLogistic (Conditional independence test for longitudinal and clustered data using GLMM), 55
testIndGLMMNB (Conditional independence test for longitudinal and clustered data using GLMM), 55
testIndGLMMNormLog, 127
testIndGLMMNormLog (Conditional independence test for longitudinal and clustered data using GLMM), 55
testIndGLMMOrdinal (Conditional independence test for longitudinal and clustered data using GLMM), 55
testIndGLMMPos, 127, 155
testIndGLMMPos (Conditional independence test for longitudinal and clustered data using GLMM), 55
testIndGLMMReg, 93, 96, 127, 129, 155, 201, 231
testIndGLMMReg (Conditional independence test for longitudinal and clustered data using GLMM), 55
testIndIGreg (Conditional independence tests for positive data), 74
testIndLM, 155
testIndLM (Conditional independence test for longitudinal and clustered data using GLMM), 55
testIndLogistic, 8, 49, 64, 73, 79, 82, 104, 113, 122, 153, 214
testIndLogistic (Conditional independence test for binary, categorical or ordinal data), 43
testIndMFMFisher (Correlation based conditional independence tests), 101
testIndMPXReg (Conditional independence tests for continuous univariate and multivariate data), 65
testIndMultinom (Conditional independence test for binary, categorical or ordinal data), 43
testIndMVreg (Conditional independence tests for continuous univariate and multivariate data), 65
testIndNB, 71, 76
testIndNB (Conditional independence tests for count data), 68
testIndNormLog (Conditional independence tests for positive data), 74
testIndOrd, 202
testIndOrd (Conditional independence test for binary, categorical or ordinal data), 43
testIndPois (Conditional independence tests for count data), 68
testIndQBinom, 45, 59, 60
testIndQBinom (Conditional independence test for binary, categorical or ordinal data), 43
testIndQPois (Conditional independence tests for count data), 68
testIndRQ, 8, 60, 68
testIndRQ (Conditional independence tests for continuous univariate and multivariate data), 65
testIndSpearman, 68, 84, 104, 198, 200
testIndSpearman (Correlation based conditional independence tests), 101
testIndSPML (Conditional independence test for circular data), 49
testIndTimeLogistic (Conditional independence test for the static-longitudinal scenario), 60
testIndTimeMultinom (Conditional independence test for the static-longitudinal scenario),
testIndTobit (Conditional independence tests for left censored data), 71

TestIndZIP, 71, 76, 177, 249

testIndZIP (Conditional independence tests for count data), 68

The max-min Markov blanket algorithm, 233

Topological sort of a DAG, 235
topological_sort, 32, 238
topological_sort (Topological sort of a DAG), 235

Total causal effect of a node on another node, 236

Transformation of a DAG into an essential graph, 238

Transitive closure of an adjacency matrix, 239

transitiveClosure (Transitive closure of an adjacency matrix), 239

triangles.search (Search for triangles in an undirected graph), 214

undir.path (Undirected path(s) between two nodes), 240

Undirected path(s) between two nodes, 240

Univariate regression based tests, 241
univariateScore (MXM-internal), 187
univregs, 41, 43, 51, 119, 120, 179
univregs (Univariate regression based tests), 241

Utilities for the skeleton of a (Bayesian) Network, 245

Variable selection using the PC-simple algorithm, 247

wald.betaregs (Many simple beta regressions), 174
wald.Internalmmpc (MXM-internal), 187
wald.Internalses (MXM-internal), 187
wald.logisticregs (Many Wald based tests for logistic and Poisson regressions with continuous predictors), 178
wald.mmpc (Constraint based feature selection algorithms), 85

wald.mmpc.path (MMPC solution paths for many combinations of hyper-parameters), 181
wald.poissonregs (Many Wald based tests for logistic and Poisson regressions with continuous predictors), 178
wald.ses (Constraint based feature selection algorithms), 85
wald.univariateScore (MXM-internal), 187
wald.univregs (Univariate regression based tests), 241
wald.zipregs (Many simple zero inflated Poisson regressions), 176
waldBeta (Conditional independence test for proportions/percentages), 58
waldBinom (Conditional independence tests for success rates), 77
waldCR (Conditional independence tests for survival data), 79
waldER (Conditional independence tests for survival data), 79
waldGamma (Conditional independence tests for positive data), 74
waldIGreg (Conditional independence tests for positive data), 74
waldLLR (Conditional independence tests for survival data), 79
waldLogistic (Conditional independence test for binary, categorical or ordinal data), 43
waldmmpc.model (Regression models based on SES and MMPC outputs), 205
waldMMReg (Conditional independence tests for continuous univariate and multivariate data), 65
waldNB (Conditional independence tests for count data), 68
waldNormLog (Conditional independence tests for positive data), 74
waldOrdinal (Conditional independence tests for binary, categorical or ordinal data), 43
waldPois (Conditional independence tests for count data), 68
waldQBinom (Conditional independence
test for binary, categorical or ordinal data, 43
waldses.model (Regression models based on SES and MMPC outputs), 205
waldTobit (Conditional independence tests for left censored data), 71
waldWR (Conditional independence tests for survival data), 79
waldZIP (Conditional independence tests for count data), 68
weibreg.mxm (Cross-Validation for SES and MMPC), 109
wr.fsreg (MXM-internal), 187
wr.fsreg_2 (MXM-internal), 187

Zero inflated Poisson and negative binomial regression, 248
zinb.mle (MXM-internal), 187
zinb.mod (Zero inflated Poisson and negative binomial regression), 248
zinb.reg (Zero inflated Poisson and negative binomial regression), 248
zip.bsreg (MXM-internal), 187
zip.fsreg (MXM-internal), 187
zip.fsreg_2 (MXM-internal), 187
zip.mod, 177
zip.mod (Zero inflated Poisson and negative binomial regression), 248
zip.reg (Zero inflated Poisson and negative binomial regression), 248
zip.regs, 249
zip.regs (Many simple zero inflated Poisson regressions), 176
zipmle.wei (MXM-internal), 187
zipwei (MXM-internal), 187