Package ‘Rfast’

February 16, 2022

Type Package
Title A Collection of Efficient and Extremely Fast R Functions
Version 2.0.6
Date 2022-02-15
Author Manos Papadakis, Michail Tsagris, Marios Dimitriadis, Stefanos Fafalios, Ioannis Tsamardinos, Matteo Fasiolo, Giorgos Borboudakis, John Burkardt, Changliang Zou, Kleanthi Laktaki and Christina Chatzipantsiou.
Maintainer Manos Papadakis <rfastofficial@gmail.com>
Depends R (>= 3.5.0), Rcpp (>= 0.12.3), RcppZiggurat
LinkingTo Rcpp (>= 0.12.3), RcppArmadillo
SystemRequirements C++17
BugReports https://github.com/RfastOfficial/Rfast/issues
URL https://github.com/RfastOfficial/Rfast
License GPL (>= 2.0)
NeedsCompilation yes
Repository CRAN
Date/Publication 2022-02-16 17:40:02 UTC

R topics documented:

  Rfast-package ..................................................... 6
  All k possible combinations from n elements .................. 10
  Analysis of covariance ......................................... 11
R topics documented:

Analysis of variance with a count variable ........................................ 12
Angular central Gaussian random values simulation .......................... 13
ANOVA for two quasi Poisson regression models ................................. 14
Apply method to Positive and Negative number ................................. 15
Apply to each column a method under condition ................................. 17
Backward selection regression ............................................................. 18
BIC (using partial correlation) forward regression ......................... 19
BIC forward regression with generalised linear models ..................... 20
Binary search algorithm .................................................................... 21
Binomial coefficient and its logarithm ................................................. 22
Bootstrap t-test for 2 independent samples ......................................... 23
Check if any column or row is fill with values ................................. 24
Check if values are integers and convert to integer ......................... 25
Check Namespace and Rd files ............................................................. 26
Check whether a square matrix is symmetric ..................................... 29
Chi-square and G-square tests of (unconditional) independence .......... 30
Cholesky decomposition of a square matrix ....................................... 31
Circular or angular regression ............................................................. 32
Circular-linear correlation ................................................................ 33
Column-wise cumulative operations (sum, prod, min, max) ............... 34
Column and row wise coefficients of variation ................................... 35
Column and row-wise Any/All ............................................................. 36
Column and row-wise means of a matrix ............................................. 37
Column and row-wise medians ........................................................... 38
Column and row-wise nth smallest value of a matrix/vector ............. 39
Column and row-wise Order - Sort Indices ........................................ 41
Column and row-wise products ......................................................... 42
Column and row-wise range of values of a matrix ............................ 43
Column and row-wise ranks ............................................................... 44
Column and row-wise Shuffle ............................................................ 45
Column and row-wise sums of a matrix ............................................. 46
Column and row-wise tabulate .......................................................... 47
Column and row-wise variances and standard deviations .................. 48
Column and rows-wise mean absolute deviations ............................. 49
Column-row wise minima and maxima of two matrices .................... 50
Column-wise differences .................................................................. 51
Column-wise kurtosis and skewness coefficients .............................. 52
Column-wise matching coefficients .................................................. 53
Column-wise minimum and maximum ............................................. 54
Column-wise MLE of some univariate distributions ......................... 55
Column-wise true/false value ............................................................ 57
Column-wise uniformity Watson test for circular data ....................... 58
Column-wise Yule’s Y (coefficient of colligation) .............................. 59
Convert a dataframe to matrix ........................................................... 60
Convert R function to the Rfast’s corresponding ......................... 61
Correlation based forward regression ............................................. 62
Correlation between pairs of variables .......................................... 63
Correlations ...................................................................................... 65
R topics documented:

Covariance and correlation matrix ........................................ 66
Cox confidence interval for the ratio of two Poisson variables .......... 67
Cross-Validation for the k-NN algorithm .................................. 68
Cross-Validation for the k-NN algorithm using the arc cosine distance . 70
Deep copy ............................................................................. 72
Density of the multivariate normal and t distributions .................... 73
Design Matrix ......................................................................... 74
Diagonal Matrix ....................................................................... 75
Distance between vectors and a matrix ...................................... 76
Distance correlation .................................................................. 77
Distance matrix ........................................................................ 78
Distance variance and covariance .............................................. 80
Eigenvalues and eigenvectors in high dimensional principal component analysis ........................................... 81
Empirical and exponential empirical likelihood tests for one sample ........ 82
Empirical and exponential empirical likelihood tests for two samples .......... 83
Energy distance between matrices .............................................. 85
Equality of objects .................................................................... 86
Estimation of an AR(1) model ..................................................... 87
Estimation of the Box-Cox transformation .................................... 88
Exact t-test for 2 independent samples ........................................ 89
Exponential empirical likelihood for a one sample mean vector hypothesis testing .............................................. 90
Exponential empirical likelihood hypothesis testing for two mean vectors .................................................................. 91
Fast and general - untyped representation of a factor variable .............. 93
FBED variable selection method using the correlation ....................... 94
Find element ............................................................................. 95
Find the given value in a hash table ............................................. 96
Fitted probabilities of the Terry-Bradley model ................................. 97
Fitting a Dirichlet distribution via Newton-Rapshon ......................... 98
Floyd-Warshall algorithm ................................................................ 99
Forward selection with generalised linear regression models .............. 101
G-square and Chi-square test of conditional independence ............... 102
Gamma regression with a log-link ............................................... 104
Gaussian regression with a log-link ............................................ 105
Generates random values from a normal and puts them in a matrix ....... 106
Get specific columns/rows fo a matrix ....................................... 107
Hash - Pair function ................................................................... 108
Hash object ............................................................................... 109
Hash object to a list object ....................................................... 110
High dimensional MCD based detection of outliers ....................... 111
Hypothesis test for the distance correlation ................................... 112
Hypothesis test for two means of percentages ................................ 114
Hypothesis test for von Mises-Fisher distribution over Kent distribution ............................................. 115
Hypothesis testing between two skewness or kurtosis coefficients ........ 116
Index of the columns of a data.frame which are a specific type .......... 117
Insert/remove function names in/from the NAMESPACE file ............... 118
Inverse Gaussian regression with a log-link ................................ 119
Inverse of a symmetric positive definite matrix .............................. 120
Iterator .................................................................................... 121
James multivariate version of the t-test .............................................. 123
k nearest neighbours algorithm (k-NN) .............................................. 124
k-NN algorithm using the arc cosinus distance ................................. 126
Limited number of eigenvalues and eigenvectors of a symmetric matrix . 127
Linear models for large scale data .................................................... 128
Logistic and Poisson regression models ............................................ 130
Logistic or Poisson regression with a single categorical predictor ....... 131
Lower and Upper triangular of a matrix ............................................ 133
Mahalanobis distance ........................................................................ 134
Many (and one) area under the curve values ..................................... 135
Many 2 sample proportions tests ....................................................... 136
Many 2 sample tests .......................................................................... 137
Many analysis of variance tests with a discrete variable .................... 139
Many ANCOVs ................................................................................. 140
Many ANOVAs for count data with Poisson or quasi Poisson models . 141
Many exponential regressions ............................................................. 142
Many F-tests with really huge matrices ............................................. 143
Many G-square and Chi-square tests of independence ....................... 144
Many Gini coefficients ...................................................................... 146
Many hypothesis tests for two means of percentages ......................... 147
Many moment and maximum likelihood estimations of variance components . 148
Many multi-sample tests ................................................................. 150
Many multivariate simple linear regressions coefficients .................... 151
Many non parametric multi-sample tests .......................................... 152
Many odds ratio tests ........................................................................ 154
Many one sample goodness of fit tests for categorical data ................ 155
Many one sample tests ..................................................................... 156
Many random intercepts LMMs for balanced data with a single identical covariate. 157
Many regression based tests for single sample repeated measures ....... 159
Many score based regressions ............................................................ 161
Many Shapiro-Francia normality tests .............................................. 163
Many simple circular or angular regressions ..................................... 164
Many simple geometric regressions .................................................... 165
Many simple linear mixed model regressions ..................................... 166
Many simple linear regressions coefficients ..................................... 167
Many simple multinomial regressions .............................................. 168
Many simple regressions for positive valued data .............................. 169
Many tests for the dispersion parameter in Poisson distribution ......... 171
Many two-way ANOVAs ................................................................. 172
Many univariate generalised linear models ...................................... 173
Many univariate simple linear regressions ....................................... 175
Many univariate simple logistic and Poisson regressions .................. 176
Many univariate simple quasi poisson regressions ............................ 178
Many Welch’s F-tests ....................................................................... 179
Match .............................................................................................. 180
Matrix multiplication ......................................................................... 181
Matrix with all pairs of t-tests .......................................................... 182
Matrix with G-square tests of independence .................................... 183
Mean - Median absolute deviation of a vector ........................................ 185
Median of a vector ................................................................. 186
Minima and maxima of two vectors/matrices .............................. 187
Minimum and maximum .......................................................... 188
Minimum and maximum frequencies ........................................ 189
MLE for multivariate discrete data ........................................ 190
MLE of (hyper-)spherical distributions .................................... 191
MLE of continuous univariate distributions defined on the positive line .......... 193
MLE of continuous univariate distributions defined on the real line .......... 195
MLE of count data (univariate discrete distributions) ..................... 196
MLE of distributions defined in the (0, 1) interval ....................... 198
MLE of some circular distributions ......................................... 200
MLE of the inverted Dirichlet distribution ................................ 201
MLE of the multivariate (log-) normal distribution ..................... 202
MLE of the multivariate t distribution .................................... 204
MLE of the ordinal model without covariates ............................ 205
MLE of the tobit model .......................................................... 206
Moment and maximum likelihood estimation of variance components ........ 207
Multi-sample tests for vectors ................................................. 209
Multinomial regression ............................................................. 211
Multivariate kurtosis ............................................................... 212
Multivariate Laplace random values simulation .......................... 213
Multivariate normal and t random values simulation .................... 214
Naive Bayes classifiers ............................................................ 215
Natural Logarithm each element of a matrix ............................. 217
Natural logarithm of the beta function ................................... 218
Natural logarithm of the gamma function and its derivatives .......... 219
Norm of a matrix .................................................................. 220
Number of equal columns between two matrices ........................ 221
Odds ratio and relative risk ....................................................... 222
One sample t-test for a vector ................................................... 223
Operations between two matrices or matrix and vector ................... 224
Orthogonal matching pursuit variable selection .......................... 226
Outer function ....................................................................... 227
Permutation ........................................................................ 228
Permutation based p-value for the Pearson correlation coefficient ....... 229
Polyserial correlation ................................................................. 230
Pooled covariance matrix ......................................................... 232
Prediction with some naive Bayes classifiers .............................. 233
Quasi binomial regression for proportions ................................ 234
Quasi Poisson regression for count data ................................... 236
Random intercepts linear mixed models .................................... 237
Random values simulation from a von Mises distribution .............. 239
Ranks of the values of a vector ................................................. 240
Reading the files of a directory ............................................... 241
Repeated measures anova ......................................................... 242
Replicate columns/rows .......................................................... 243
Representation of Stack ......................................................... 244
A collection of Rfast functions for data analysis. Note 1: The vast majority of the functions accept matrices only, not data.frames. Note 2: Do not have matrices or vectors with have missing data (i.e NAs). We do no check about them and C++ internally transforms them into zeros (0), so you may get wrong results. Note 3: In general, make sure you give the correct input, in order to get the correct output. We do no checks and this is one of the many reasons we are fast.
Details
Maintainers

Manos Papadakis <papadakm95@gmail.com>

Note

Acknowledgments: We would like to acknowledge:

• Professor Kurt Hornik, Doctor Uwe Ligges (and the rest of R core team) for their invaluable help with this R package.
• Erich Studerus for his invaluable comments.
• Neal Fultz for his suggestions.
• Vassilis Vasdekis for his invaluable help with the random effects models.
• Marios Dimitriadis’ work was funded by the Special Account for Research Funds of the University of Crete, Department of Computer Science.
• Phillip Si is greatly acknowledged for his help with the Floyd-Warshal algorithm.
• Keefe Murphy for his invaluable help with NEWS file and for his suggestions.
• Zacharias Papadovassilakis gave us the inspiration for the memory efficient version of the k-NN algorithm.
• Yannis Pantazis explained us how the orthogonal matching pursuit works.
• Achim Zeileis for his help with the quasi Poisson regression models.
• Pedro J. Aphalo for finding a bug.
• Dimitris Kyriakis for finding a bug.
• Cyrille Conord for finding a bug.
• Aaron Robotham for finding a bug.
• Calvin Pan from the Department of Human Genetics at UCLA found a bug in the function glm_logistic and he is greatly acknowledged for that.
• Adam Muschielok from Rodenstock GmbH found a bug in the function vmf.mle and he is greatly acknowledged for that.
• Bret Presnell for detecting and correcting a bug in the function rvmf.
• Gabriel Hoffman for spotting a mistake in the function dirimultinom.mle.

From now on the Rfast can be used in C++ via LinkingTo mechanism.
The main namespace is "Rfast". Inside "Rfast" you will find two more namespaces, "vector" and "matrix".

Namespace "vector" for calling functions using an Rcpp’s or RcppArmadillo’s vector.

Namespace "matrix" for calling functions using an Rcpp’s or RcppArmadillo’s matrices.

The signatures of the functions and the arguments are the same that are exported in R.

For namespace "vector" the functions that are available are:

1. median(x)
2. var(x, std = false, na_rm = false)
3. mad(x, method = "median", na_rm = false)
4. shuffle(x, engine = Engine(time(0)) // Engine by default is default_random_engine. You can use anyone from C++.

For namespace "matrix" the functions that are available are:

1. transpose(x)
2. matrix_multiplication(x, y)
3. colSort(x, descend = false, stable = false, parallel = false)
4. rowSort(x, descend = false, stable = false, parallel = false)
5. is_symmetric(x)
6. colMedian(x, na_rm = false, parallel = false)
7. rowMedian(x, na_rm = false, parallel = false)
8. colVars(x, std = false, na_rm = false, parallel = false)
9. rowVars(x, std = false, na_rm = false, parallel = false)
10. colMads(x, method = "median", na_rm = false, parallel = false)
11. rowMads(x, method = "median", na_rm = false, parallel = false)
12. colShuffle(x, engine = Engine(time(0)) // Engine by default is default_random_engine. You can use anyone from C++.
13. rowShuffle(x, engine = Engine(time(0)) // Engine by default is default_random_engine. You can use anyone from C++.

How to use it:

1. Just add in "LinkingTo" in your NAMESPACE file the "Rfast" or in Rstudio "[[Rcpp::depends(Rfast)]]."
2. Include in your cpp files the header "Rfast.h" and enjoy!

Author(s)

- Manos Papadakis <papadakm95@gmail.com>
- Michail Tsagris <mtsagris@uoc.gr>
- Marios Dimitriadis <kmdimitriadis@gmail.com>
- Stefanos Fafalios <stefanosfafalios@gmail.com>
- Ioannis Tsamardinos <tsamard@csd.uoc.gr>
All k possible combinations from n elements

- Matteo Fasiolo <matteo.fasiolo@gmail.com>
- Giorgos Borboudakis <borbudak@gmail.com>
- John Burkardt <jburkardt@fsu.edu>
- Kleanthi Lakiotaki <kliolak@gmail.com>
- Changliang Zou <nk.chlzou@gmail.com>
- Christina Chatzipantsiou <chatzipantsiou@gmail.com>

All k possible combinations from n elements

Description

All k possible combinations from n elements.

Usage

`comb_n(n, k, simplify=TRUE)`

Arguments

- `n` A positive INTEGER number or a vector with numbers.
- `k` A positive integer number at most equal to n or at most equal to the length of n, if n is a vector.
- `simplify` A logical value for return List instead of matrix.

Value

A matrix with k columns and rows equal to the number of possible unique combinations of n with k elements. If simplify is set to TRUE then a list with k values where each value has length equal to the number of possible unique combinations of n with k elements.

Author(s)

Manos Papadakis and Marios Dimitriadis

R implementation and documentation: Manos Papadakis <papadakm95@gmail.com> and Marios Dimitriadis <kmdimitriadis@gmail.com>.

References


See Also

`nth, colMaxs, colMins, colrange`
Examples

```
system.time( comb_n(20, 4) )
system.time( combn(20, 4) )
x <- rnorm(5)
res<-comb_n(x, 3)
```

---

### Analysis of covariance

#### Description

Analysis of covariance

#### Usage

```
ancova1(y, ina, x, logged = FALSE)
```

#### Arguments

- `y`: A numerical vector with the data, the response variable.
- `ina`: A numerical vector with 1s, 2s, 3s and so one indicating the two groups. Be careful, the function is designed to accept numbers greater than zero.
- `x`: A numerical vector whose length is equal to the number of rows of `y`. This is the covariate.
- `logged`: Should the p-values be returned (FALSE) or their logarithm (TRUE)?

#### Details

Analysis of covariance is performed. No interaction between the factor and the covariate is tested. Only the main effects. The design need not be balanced. The values of `ina` need not have the same frequency. The sums of squares have been adjusted to accept balanced and unbalanced designs.

#### Value

A matrix with the test statistic and the p-value for the factor variable and the covariate.

#### Author(s)

Michail Tsagris

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

#### References

Analysis of variance with a count variable

See Also
ancovas, ftests, ttests, anova1

Examples

```r
y <- rnorm(90)
inga <- rbinom(90, 2, 0.5) + 1
x <- rnorm(90)
system.time( a <- ancova1(y, ina, x) )
```

Description
Analysis of variance with a count variable.

Usage

```
poisson.anova(y, ina, logged = FALSE)
geom.anova(y, ina, type = 1, logged = FALSE)
quasipoisson.anova(y, ina, logged = FALSE)
```

Arguments

- **y**: A numerical vector with discrete valued data, i.e. counts.
- **ina**: A numerical vector with discrete numbers starting from 1, i.e. 1, 2, 3, 4,... or a factor variable. This is suppose to be a categorical predictor. If you supply a continuous valued vector the function will obviously provide wrong results.
- **type**: This argument is for the geometric distribution. Type 1 refers to the case where the minimum is zero and type 2 for the case of the minimum being 1.
- **logged**: Should the p-values be returned (FALSE) or their logarithm (TRUE)?

Details
This is the analysis of variance with Poisson or geometric distributed data. What we do is a log-likelihood ratio test. However, this is exactly the same as Poisson regression with a single predictor variable who happens to be categorical. Needless to say that this is faster function than the glm command in R. For the same purpose with a Bernoulli variable use g2Test. The quasipoisson.anova is when in the glm function you specify family = quasipoisson. This is suitable for the case of over or under-dispersed data.

Value
A vector with two values, the difference in the deviances (or the scale difference in the case of quasi poisson) and the relevant p-value. The quasipoisson.anova also returns the estimate of the \( \phi \) parameter.
Angular central Gaussian random values simulation

Author(s)

Michail Tsagris

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

See Also

logistic.cat1,g2Test,poisson.anovas,anova,poisson_only,poisson.mle

Examples

```r
y <- rpois(300, 10)
inga <- rbinom(300, 3, 0.5) + 1
a1 <- poisson.anova(y, ina)
a2 <- glm(y ~ ina, poisson)

## Not run:
res<-anova(a2, test = "Chisq")
## End(Not run)
y <- rgeom(300, 0.7)
res<-geom.anova(y, ina)
```

Description

Angular central Gaussian random values simulation.

Usage

```r
racg(n, sigma, seed = NULL)
```

Arguments

- `n`: The sample size, a numerical value.
- `sigma`: The covariance matrix in \( \mathbb{R}^d \).
- `seed`: If you want the same to be generated again use a seed for the generator, an integer number.

Details

The algorithm uses univariate normal random values and transforms them to multivariate via a spectral decomposition. The vectors are then scaled to have unit length.
ANOVA for two quasi Poisson regression models

Value
A matrix with the simulated data.

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

References

See Also
acg.mle, rmvnorm, rmvlaplace, rmvt

Examples
s <- cov(iris[,1:4])
x <- racg(100, s)
res<-acg.mle(x)
res<-vmf.mle(x) # the concentration parameter, kappa, is very low, close to zero, as expected.

Description
ANOVA for two quasi Poisson regression models.

Usage
anova_quasipois.reg(mod0, mod1, n)

Arguments
mod0 An object as returned by the "qpois.reg" function. This is the null model.
mod1 An object as returned by the "qpois.reg" function. This is the alternative model.
n The sample size. This is necessary to calculate the degrees of freedom.

Details
This is an ANOVA type significance testing for two quasi Poisson models.
Value

A vector with 4 elements, the test statistic value, its associated p-value and the relevant degrees of freedom of the numerator and the denominator.

Author(s)

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

References


See Also

`anova_qpois.reg, qpois.reg, univglm, quasipoisson.anova`

Examples

```r
# Not run:
y <- rnbinom(200, 10, 0.5)
x <- matrix(rnorm(200 * 3), ncol = 3)
a1 <- qpois.reg(x, y)
a0 <- qpois.reg(x[, 1], y)
res <- anova_quasipois.reg(a0, a1, 200)
b1 <- glm(y ~ x, family = quasipoisson)
b0 <- glm(y ~ x[, 1], family = quasipoisson)
res <- anova(b0, b1, test = "F")
c1 <- glm(y ~ x, family = poisson)
c0 <- glm(y ~ x[, 1], family = poisson)
res <- anova(c0, c1, test = "Chisq")
# End(Not run)
```

Description

Apply method to Positive and Negative number.
Apply method to Positive and Negative number

Usage

negative(x, method = "min")
positive(x, method = "min")
positive.negative(x, method = "min")

Arguments

x A numerical vector with data.
method Accept 3 values. "min", "max", "min.max".

Details

These functions apply the chosen method to the chosen subset (negative, positive, or both) from the vector and return the result.

Value

negative: apply the chosen method to every negative number of the input vector. positive: apply the chosen method to every positive number of the input vector. positive.negative: apply the chosen method to every negative and positive number of the input vector.

Author(s)

Manos Papadakis

R implementation and documentation: Manos Papadakis <papadakm95@gmail.com>.

See Also

nth, colnth, rownth, sort_unique, Round

Examples

x <- rnorm(1000)
identical(negative(x, "min"), min(x<0))
identical(positive(x, "min"), min(x>0))
identical(positive.negative(x, "min"), c(min(x<0), min(x>0)))

x<-NULL
Apply to each column a method under condition

Description

Apply to each column a method under condition.

Usage

apply.condition(x, method = "+", oper = ">", cond.val = 0)

Arguments

- **x**: An integer matrix.
- **method**: One of: "+", "-", "+", "+", "min", "max".
- **oper**: One of: ">", ">", ">", ">", ">", ">".
- **cond.val**: An integer value for the condition.

Details

Apply to each col the specified method using the condition.

Value

An integer vector with the corresponding values.

Author(s)

Manos Papadakis and Michail Tsagris

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

See Also

colsums, colMedians, colVars

Examples

```r
x <- matrix(rpois(100, 6), 10, 10)
identical(apply(x, 2, function(x) { sum(x > 0) }), apply.condition(x, "+", ">", 0))
x<-NULL
```
Backward selection regression

Description

Backward selection regression.

Usage

bs.reg(y, x, alpha = 0.05, type = "logistic")

Arguments

y A numerical vector with the response variable values. It can either be of 0 and 1 values (Logistic regression) or of integer values 0, 1, 2,... (Poisson regression).

x A numerical matrix with the candidate variables.

alpha Threshold (suitable values are in [0,1]) for assessing the significance of p-values. The default value is at 0.05.

type For the Logistic regression put "logistic" (default value) and for Poisson type "poisson".

Details

This function currently implements only the binary Logistic and Poisson regressions. If the sample size is less than the number of variables a notification message will appear and no backward regression will be performed.

Value

The output of the algorithm is an S3 object including:

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

Vars A vector with the selected variables.

Author(s)

Marios Dimitriadis

R implementation and documentation: Marios Dimitriadis <mtsagris@csd.uoc.gr>

See Also

fs.reg,univglm,cor.fsreg
Examples

```r
y <- rbinom(50, 1, 0.5)
x <- matrnorm(50, 10)
res <- bs.reg(y, x)
```

Description

BIC (using partial correlation) forward regression.

Usage

```r
bic.corfsreg(y, x, tol = 2)
```

Arguments

- `y`: A numerical vector.
- `x`: A matrix with data, the predictor variables.
- `tol`: If the BIC difference between two successive models is less than the tolerance value, the variable will not enter the model.

Details

The forward regression tries one by one the variables using the F-test, basically partial F-test every time for the latest variable. This is the same as testing the significance of the coefficient of this latest entered variable. Alternatively the correlation can be used and this case the partial correlation coefficient. There is a direct relationship between the t-test statistic and the partial correlation coefficient. Now, instead of having to calculate the test statistic, we calculate the partial correlation coefficient. The largest partial correlation indicates the candidate variable to enter the model. If the BIC of the regression model with that variable included, reduces, less than "tol" from the previous model without this variable, the variable enters.

Value

A matrix with two columns, the index of the selected variable(s) and the BIC of each model. The first line is always 0 and the BIC of the model with no predictor variables.

Author(s)

Michail Tsagris

R implementation and documentation: Michail Tsagris <mtsagris@uoc.gr> and Manos Papadakis <papadakm95@gmail.com>.
References


See Also
cor.fsreg, score.glms, univglms, logistic_only, poisson_only, regression

Examples

## 200 variables, hence 200 univariate regressions are to be fitted
x <- matrix(rnorm(200 * 200), ncol = 200)
y <- rnorm(200)
system.time(a1 <- bic.corfsreg(y, x))
system.time(a2 <- cor.fsreg(y, x))
x <- NULL

BIC forward regression with generalised linear models

**BIC forward regression with generalised linear models**

Description

BIC forward regression with generalised linear models.

Usage

bic.fs.reg(y, x, tol = 2, type = "logistic")

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>y</td>
<td>A numerical vector.</td>
</tr>
<tr>
<td>x</td>
<td>A matrix with data, the predictor variables.</td>
</tr>
<tr>
<td>tol</td>
<td>If the BIC difference between two successive models is less than the tolerance value, the variable will not enter the model.</td>
</tr>
<tr>
<td>type</td>
<td>If you have a binary dependent variable, put &quot;logistic&quot;. If you have count data, put &quot;poisson&quot;.</td>
</tr>
</tbody>
</table>

Details

The forward regression tries one by one the variables using the BIC at each step for the latest variable. If the BIC of the regression model with that variable included, is less than "tol" from the previous model without this variable, the variable enters.

Value

A matrix with two columns, the index of the selected variable(s) and the BIC of each model.
Author(s)

Marios Dimitriadis
R implementation and documentation: Marios Dimitriadis <kmdimitriadis@gmail.com>.

References


See Also

fs.reg, bic.corfsreg, cor.fsreg, score.glms, univglms, logistic_only, poisson_only, regression

Examples

```r
## Not run:
x <- matrix(rnorm(200 * 50), ncol = 50)
## 200 variables, hence 200 univariate regressions are to be fitted
y <- rbinom(200, 1, 0.5)
a <- bic.fs.reg(y, x)
x <- NULL
## End(Not run)
```

---

**Binary search algorithm**

*Binary search algorithm*

---

**Description**

Search a value in an ordered vector.

**Usage**

```r
binary_search(x, v, index=FALSE)
```

**Arguments**

- `x`: A vector with the data.
- `v`: A value to check if exists in the vector `x`.
- `index`: A boolean value for choose to return the position inside the vector.

**Details**

The functions is written in C++ in order to be as fast as possible.

**Value**

Search if the `v` exists in `x`. Then returns TRUE/FALSE if the value is been found.
Binomial coefficient and its logarithm

Author(s)

Manos Papadakis

R implementation and documentation: Manos Papadakis <papadakm95@gmail.com>.

See Also

is_element

Examples

```r
x <- sort(rnorm(1000))
v <- x[50]
b <- binary_search(x,v)
b1 <- binary_search(x,v,TRUE)
```

Description

Binomial coefficient and its logarithm.

Usage

```r
Lchoose(x, k)
Choose(x, k)
```

Arguments

- `x`: A vector with integer values numbers.
- `k`: A positive non zero at most equal to x.

Details

The binomial coefficient or its logarithm are evaluated.

Value

A vector with the answers.

Author(s)

Manos Papadakis

R implementation and documentation: Michail Tsagris <mtsagris@uoc.gr>
**Bootstrap t-test for 2 independent samples**

**Description**

Bootstrap t-test for 2 independent samples.

**Usage**

```r
boot.ttest2(x, y, B = 999)
```

**Arguments**

- `x`: A numerical vector with the data.
- `y`: A numerical vector with the data.
- `B`: The number of bootstrap samples to use.

**Details**

Instead of sampling B times from each sample, we sample $\sqrt{B}$ from each of them and then take all pairs. Each bootstrap sample is independent of each other, hence there is no violation of the theory.

**Value**

A vector with the test statistic and the bootstrap p-value.

**Author(s)**

Michail Tsagris and Christina Chatzipantsiou

R implementation and documentation: Michail Tsagris <mtsagris@uoc.gr> and Christina Chatzipantsiou <chatzipantsiou@gmail.com>.

---

**Examples**

```r
x <- sample(20:30, 100, replace = TRUE)
res <- Choose(x, 4)
res <- Lchoose(x, 4)

x <- NULL
```
References


See Also
ttest2,exact.ttest2,ftest

Examples

tic <- proc.time()
x <- rexp(40, 4)
y <- rbeta(50, 2.5, 7.5)
system.time( a <- boot.ttest2(x, y, 9999) )
a

Check if any column or row is fill with values

Description

Check if any column or row is fill with values.

Usage

colrow.value(x,value=0)

Arguments

x A vector with data.
value A value to check.

Details

Check all the column if any has all its elements equal to argument value. If found, return "TRUE". Otherwise continues with rows. If columns and rows hasn’t any value vector then return "FALSE". Even if it returns "FALSE" that doesn’t mean the determinant can’t be value. It might be but if check before and found any value vector then for sure the determinant it’ll be value.
Value
A boolean value, "TRUE" if any column OR row is all filled with value. "FALSE" otherwise.

Author(s)
Manos Papadakis
R implementation and documentation: Manos Papadakis <papadakm95@gmail.com>.

See Also
rowMins,rowFalse,nth,colrange,colMedians,colVars,colSort,rowSort,rowTrue

Examples
```r
x <- matrix(runif(10*10),10,10)
res<-colrow.value(x)
x<-NULL
```

Description
Check if values are integers and convert to integer.

Usage
```r
is_integer(x)
as_integer(x,result.sort = TRUE,init = 1)
```

Arguments
- `x`: is_integer: A vector with numeric data. as_integer: A vector with data.
- `result.sort`: A logical value for sorting the result.
- `init`: An integer value to start.

Details
The behavior of these functions are different than R’s built in.

is_integer: check if all the values are integers in memory. If typeof is double, and the values are integers in range \(-2^{31} : 2^{31}\) then it is better to convert to integer vector for using less memory. Also you can decrease the time complexity.
as_integer: converts the discrete values to integers.
Check Namespace and Rd files

Description

Check Namespace/Rd and examples files.

Value

is_integer: A logical value, TRUE if all values are integers and in range \(-2^{31} : 2^{31}\). Otherwise FALSE.

as_integer: By default the function will return the same result with "as.numeric" but the user can change the "init" value not start from 1 like R's. Also the result can be unsorted using "result.sort".

Author(s)

R implementation and documentation: Manos Papadakis <papadakm95@gmail.com>.

See Also

as_integer, colVars, colmeans, read.directory

Examples

```r
x<-runif(10)
y1<-is_integer(x) # y1 is FALSE
x<-as.numeric(rpois(10,10)) # integers but typeof is double
y1<-is_integer(x) # y1 is TRUE so you can convert to integer vector.

as_integer(letters) ## as.numeric(letters) produce errors
x<-y1<-NULL
```

Check Namespace and Rd files

Check Namespace and Rd files

Description

Check Namespace/Rd and examples files.

Usage

```r
checkNamespace(path.namespace,path.rfolder)
checkAliases(path.man,path.rfolder)
checkTF(path.man)
checkExamples(path.man,each = 1,print.errors = stderr(),
print.names = FALSE)
checkUsage(path.man,path.rfolder)
```

Arguments

- `path.namespace`: An full path to the "NAMESPACE" file.
- `path.rfolder`: An full path to the directory that contains the "R" files.
- `path.man`: An full path to the directory that contains the "Rd" files.
Check Namespace and Rd files

For function "checkNamespace": reads from the NAMESPACE folder all the export R functions, reads from folder R all the R functions and check if all the functions are export.

For function "checkAliases": reads from the man directory all the Rd files, then reads from each file the aliases and check if: 1) All the R files has man file or an alias. 2) All aliases belongs to functions. 3) If there are dublicated aliases.

For function "checkExamples": reads from the man directory all the Rd files, then read from each file the examples and then run each of them. If you want to print the errors in any file then set "print.errors=file_name" or in the standard error "print.errors=stderr()" and then you will see all the errors for every file. For succeed run of your code you should first run "library(PACKAGE_NAME)". The argument "print.names" it is very helpful because if any of you function crashes R during running you will never know which one was. So setting it "TRUE", it will print the name of each file before runnign it's example. It might crash, but you will know which file. Remember that there is always an error timeout so it might didn’t crash the current file but one from the previous.

For function checkTF: reads from the man directory all the Rd files, then read from each file the examples and checks if any examples has the values "T" and "F" instead "TRUE" and "FALSE". The "T","F" is wrong.

For function checkUsage: reads from the man directory all the Rd files and for each man check if the usage section has the right signature for the functions from the R directory.

For functions "checkTF", "checkUsage", "checkAliases" you can choose which files not to read for both R and Rd. You must add in the first line of the file in comment the "attribute" "[dont read]". Then each function will now which file to read or not. For Rd you add "%[dont read]" and for R "#[dont read]". Finally, these functions will return in the result a list of which files had this attribute.

Value

For function "checkNamespace": a vector with the names of missing R files. (Don’t use it for now)

For function "checkAliases": a list with 4 fields.

- Missing Man files: A vector with the names of the missing Rd files or nothing.
- Missing R files: A vector with the names of the missing R files or nothing.
- Duplicate alias: A vector with the names of the dublicate aliases or nothing.
- dont read: A list with 2 fields: R: A character vector whith the names of the files that had attribute "#[dont read]" or nothing. Rd: A character vector whith the names of the files that had attribute "%[dont read]" or nothing.

For function "checkExamples": a list with 3 fields

- Errors: A character vector with the names of the Rd files that produced an error.
Check Namespace and Rd files

Big Examples
A character vector with the names of the Rd files that has big examples per line.
dont read
A list with 2 fields R: A character vector with the names of the files that had attribute "#[dont read]" or nothing. Rd: A character vector with the names of the files that had attribute "%[dont read]" or nothing.

For function "checkTF": a list with 3 fields
TRUE
A character vector with the names of the Rd files that has "T" or nothing.
FALSE
A character vector with the names of the Rd files that has "F" or nothing.
dont read
A list with 2 fields R: A character vector with the names of the files that had attribute "#[dont read]" or nothing. Rd: A character vector with the names of the files that had attribute "%[dont read]" or nothing.

For function "checkUsage": a list with 3 fields
missing functions
A character vector with the name of the file that is missing and the Rd file that is found or nothing.
mismatch functions
A character vector with the name of the file that has mismatch function and the Rd file that is found or nothing.
dont read
A list with 2 fields R: A character vector with the names of the files that had attribute "#[dont read]" or nothing. Rd: A character vector with the names of the files that had attribute "%[dont read]" or nothing.

Author(s)
R implementation and documentation: Manos Papadakis <papadakm95@gmail.com>.

See Also
read.directory, AddToNamespace, sourceR, sourceRd, read.examples

Examples
## Not run:
for example: path.namespace="C:\some_file\NAMESPACE"
for example: path.rfolder="C:\some_file\R"
for example: path.man="C:\some_file\man"
system.time( a<-checkNamespace(path.namespace,path.rfolder) )
system.time( b<-checkAliases(path.man,path.rfolder) )
system.time( b<-checkExamples(path.man) )
system.time( b<-checkExamples(path.man,2) )
system.time( b<-checkTF(path.man) )
system.time( b<-checkTF(path.man,path.rfolder) )
## End(Not run)
Description

Check whether a square matrix is symmetric.

Usage

is.symmetric(x)

Arguments

x

A square matrix with data.

Details

Instead of going through the whole matrix, the function will stop if the first disagreement is met.

Value

A boolean value, TRUE or FALSE.

Author(s)

Manos Papadakis

R implementation and documentation: Manos Papadakis <papadakm95@gmail.com>.

See Also

cholesky, cora, cova

Examples

x <- matrix(rnorm(100 * 400), ncol = 400)
s1 <- cor(x)
is.symmetric(s1)
x <- x[1:100, ]
is.symmetric(x)
x <- s1 <- NULL
Chi-square and G-square tests of (unconditional) independence

Description

Chi-square and G-square tests of (unconditional) independence.

Usage

gchi2Test(x, y, logged = FALSE)

Arguments

x
A numerical vector or a factor variable with data. The data must be consecutive numbers.

y
A numerical vector or a factor variable with data. The data must be consecutive numbers.

logged
Should the p-values be returned (FALSE) or their logarithm (TRUE)?

Details

The function calculates the test statistic of the $\chi^2$ and the $G^2$ tests of unconditional independence between x and y. x and y need not be numerical vectors like in g2Test. This function is more close to the spirit of MASS’ loglm function which calculates both statistics using Poisson log-linear models (Tsagris, 2017).

Value

A matrix with two rows. In each row the X2 or G2 test statistic, its p-value and the degrees of freedom are returned.

Author(s)

Manos Papadakis and Michail Tsagris

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

References


See Also

g2Test_univariate,g2Test_univariate_perm,g2Test
Cholesky decomposition of a square matrix

Examples

```r
nvalues <- 3
nvars <- 2
nsamples <- 5000
data <- matrix( sample( 0:(nvalues - 1), nvars * nsamples, replace = TRUE ), nsamples, nvars )

res<-gchi2Test(data[, 1], data[, 2])
res<-g2Test_univariate( data, rep(3, 2) ) ## G^2 test
res<-chisq.test(data[, 1], data[, 2]) ## X^2 test from R

data<-NULL
```

---

Cholesky decomposition of a square matrix

**Description**

Cholesky decomposition of a square matrix.

**Usage**

```r
cholesky(x,parallel = FALSE)
```

**Arguments**

- `x`: A square positive definite matrix.
- `parallel`: A boolean value for parallel version.

**Details**

The Cholesky decomposition of a square positive definite matrix is computed. The use of parallel is suggested for matrices with dimensions of 1000 or more.

**Value**

An upper triangular matrix.

**Author(s)**

Manos Papadakis

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

**See Also**

`is.symmetric`
Circular or angular regression

Examples

```r
x = matrix(rnorm(1000 * 50), ncol = 50)
s = cov(x)
system.time(a1 <- cholesky(s))
system.time(a2 <- chol(s))
all.equal(a1[upper.tri(a1)], a2[upper.tri(a2)])
x <- x
s <- s
a1 <- a1
a2 <- a2
```

Circular or angular regression

Description

Regression with circular dependent variable and Euclidean or categorical independent variables.

Usage

```r
spml.reg(y, x, tol = 1e-07, seb = FALSE, maxiters = 100)
```

Arguments

- **y**: The dependent variable, it can be a numerical vector with data expressed in radians or it can be a matrix with two columns, the cosinus and the sinus of the circular data. The benefit of the matrix is that if the function is to be called multiple times with the same response, there is no need to transform the vector every time into a matrix.
- **x**: The independent variable(s). Can be Euclidean or categorical (factor variables).
- **tol**: The tolerance value to terminate the Newton-Raphson algorithm.
- **seb**: Do you want the standard error of the estimates to be returned? TRUE or FALSE.
- **maxiters**: The maximum number of iterations to implement.

Details

The Newton-Raphson algorithm is fitted in this regression as described in Presnell et al. (1998).

Value

A list including:

- **iters**: The number of iterations required until convergence of the EM algorithm.
- **be**: The regression coefficients.
Circular-linear correlation

seb The standard errors of the coefficients.
loglik The value of the maximised log-likelihood.
seb The covariance matrix of the beta values.

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

References

See Also
spml.mle, iag.mle, acg.mle

Examples
## Not run:
x <- rnorm(100)
z <- cbind(3 + 2 * x, 1 - 3 * x)
y <- cbind( rnorm(100, z[,1], 1), rnorm(100, z[,2], 1) )
y <- y / sqrt( rowsums(y^2) )
a1 <- spml.reg(y, x)
y <- atan( y[,2] / y[,1] ) + pi * I(y[,1] < 0)
a2 <- spml.reg(y, x)
## End(Not run)

Circular-linear correlation

Circular-linear correlation

Description
It calculates the squared correlation between a circular and one or more linear variables.

Usage
circlin.cor(theta, x)

Arguments
theta A circular variable expressed in radians.
x The linear variable or a matrix containing many linear variables.
Details
The squared correlation between a circular and one or more linear variables is calculated.

Value
A matrix with as many rows as linear variables including:
- R-squared: The value of the squared correlation.
- p-value: The p-value of the zero correlation hypothesis testing.

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

References

See Also
spml.reg

Examples
phi <- rvonmises(50, 2, 20, rads = TRUE)
x <- 2 * phi + rnorm(50)
y <- matrix(rnorm(50 * 5), ncol = 5)
res<-circlin.cor(phi, x)
res<-circlin.cor(phi, y)
y <- NULL

Colum-wise cumulative operations (sum, prod, min, max)

Description
Colum-wise cumulative operations (sum, prod, min, max).

Usage
colCumSums(x)
colCumProds(x)
colCumMins(x)
colCumMaxs(x)
Arguments

x  A numerical matrix.

Details

Cumulative mins, maxs, sums and prods are returned.

Value

A matrix with the results. It has one row less than the initial matrix.

Author(s)

Manos Papadakis and Michail Tsagris

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

See Also

colsums, colMedians, colVars

Examples

x <- matrnorm(10, 10)
res<-colCumSums(x)
res<-colCumMins(x)
res<-colCumMaxs(x)
res<-colCumProds(x)

Column and row wise coefficients of variation

Description

Column and row wise coefficients of variation.

Usage

colcvs(x, ln = FALSE, unbiased = FALSE)
rowcvs(x, ln = FALSE, unbiased = FALSE)

Arguments

x  A numerical matrix with the data.
ln  If you have log-normally distributed data (or assume you do), then set this to TRUE.
unbiased  A boolean variable indicating whether the unbiased formula should be returned. This is applicable in case of small samples.
Details

The column-wise coefficients of variation are calculated.

Value

A vector with the coefficient of variation for each column or row.

Author(s)

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

See Also

colsums, colVars

Examples

m <- rnorm(100, 10)
x <- matrix(rnorm(100 * 100, m, 1), ncol = 100)
a1 <- colcvs(x)
a2 <- colcvs(x[1:25, ], unbiased = TRUE)
a3 <- colcvs(exp(x), ln = TRUE)
x <- NULL
Column and row-wise means of a matrix

Details

The functions is written in C++ in order to be as fast as possible.

Value

A vector where item "i" is true if found Any/All true in column/row "i". Otherwise false.

Author(s)

R implementation and documentation: Manos Papadakis <papadakm95@gmail.com>.

See Also

Median, colMedians, colMeans (built-in R function)

Examples

```r
x <- matrix(as.logical(rbinom(100*100,1,0.5)),100,100)
system.time( a<-colAny(x) )
system.time( b<-apply(x,2,any) )
all.equal(a,b)

system.time( a<-rowAny(x) )
system.time( b<-apply(x,1,any) )
all.equal(a,b)

system.time( a<-colAll(x) )
system.time( b<-apply(x,2,all) )
all.equal(a,b)

a<b<-x<-NULL
```

Description

Column and row-wise means of a matrix.

Usage

```r
colmeans(x, parallel = FALSE)
rowmeans(x)
colhameans(x, parallel = FALSE)
rowhameans(x)
```
Column and row-wise medians

Arguments

x A numerical matrix with data.

parallel Do you want to do it in parallel in C++? TRUE or FALSE.

Value

A vector with the column or row arithmetic or harmonic means.

Author(s)

Manos Papadakis

R implementation and documentation: Manos Papadakis <papadakm95@gmail.com>.

See Also

colsums, rowsums, colMins, colMedians, colMads

Examples

x <- matrix(rpois(100 * 100, 10), ncol = 100)
x1 <- colmeans(x)
x2 <- colMeans(x)
all.equal(x1, x2)

x1 <- rowmeans(x)
x2 <- rowMeans(x)
all.equal(x1, x2)

system.time( colhameans(x) )

system.time( rowhameans(x) )

x <- x1 <- x2 <- NULL

Column and row-wise medians

Description

Column and row-wise medians of a matrix.

Usage

colMedians(x, na.rm = FALSE, parallel = FALSE)

rowMedians(x, na.rm = FALSE, parallel = FALSE)
Column and row-wise nth smallest value of a matrix/vector

Arguments

\texttt{x} \quad \text{A matrix with the data.}

\texttt{parallel} \quad \text{Do you want to do it in parallel in C++? TRUE or FALSE.}

\texttt{na.rm} \quad \text{TRUE or FALSE for remove NAs if exists.}

Details

The functions is written in C++ in order to be as fast as possible.

Value

A vector with the column medians.

Author(s)

R implementation and documentation: Manos Papadakis <papadakm95@gmail.com>.

See Also

\texttt{Median, colVars, colMeans} (built-in R function)

Examples

\begin{verbatim}
x <- matrix( rnorm(100 * 100), ncol = 100 )
a <- apply(x, 2, median)
b1 <- colMedians(x)
all.equal(as.vector(a), b1)
\end{verbatim}

\begin{verbatim}
null
\end{verbatim}

Description

Column and row-wise nth smallest value of a matrix/vector.

Usage

\begin{verbatim}
colnth(x, elems, num.of.nths = 1, descending = FALSE, na.rm = FALSE, index.return = FALSE, parallel = FALSE)
rownth(x, elems, num.of.nths = 1, descending = FALSE, na.rm = FALSE, index.return = FALSE, parallel = FALSE)
nth(x, k, num.of.nths = 1, descending = FALSE, index.return = FALSE, na.rm = FALSE)
\end{verbatim}
Arguments

- **x**: A matrix with the data.
- **elems**: An integer vector with the kth smallest number to be returned for each column/row.
- **k**: The kth smallest/biggest number to be returned.
- **num.of.nths**: The number of the returned nths. By default is 1. Not use with argument parallel, for now.
- **descending**: A boolean value (TRUE/FALSE) for descending order (biggest number). By default is ascending (smallest number).
- **index.return**: Return the index of the kth smallest/biggest number.
- **parallel**: Do you want to do it in parallel in C++? TRUE or FALSE only for col-row wise.
- **na.rm**: TRUE or FALSE for remove NAs if exists. Only for function "nth".

Details

The functions is written in C++ in order to be as fast as possible.

Value

For "colnth", "rownth": A vector with the column/row nth
For "nth": The nth value.

Author(s)

Manos Papadakis <papadakm95@gmail.com>

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

See Also

- **Median, colMedians, colMeans** (built-in R function)

Examples

```r
x <- matrix(rnorm(100 * 100), ncol = 100)
elems <- sample(1:100, 100, TRUE)
system.time(colnth(x, elems) )
system.time(rownth(x, elems) )
x <- rnorm(1000)
nth(x, 500)
sort(x)[500]
```

x<-elems<-NULL
Description

Column and row-wise Order - Sort Indices.

Usage

colOrder(x, stable=FALSE, descending=FALSE, parallel = FALSE)
rowOrder(x, stable=FALSE, descending=FALSE, parallel = FALSE)
Order(x, stable=FALSE, descending=FALSE, partial = NULL)

Arguments

x A matrix with numbers or a numeric/character vector.
stable A boolean value for using a stable sorting algorithm.
descending A boolean value (TRUE/FALSE) for sorting the vector in descending order. By default sorts the vector in ascending.
parallel A boolean value for parallel version.
partial A boolean value for partial sorting.

Details

The function applies "order" in a column or row-wise fashion or Order a vector. If you want the same results as R’s, then set "stable=TRUE" because "stable=FALSE" uses a sorting algorithm that it is not stable like R’s sort. But it is faster to use the default. This version is faster for large data, more than 300.

Value

For "colOrder" and "rowOrder" a matrix with integer numbers. The result is the same as apply(x, 2, order) or apply(x, 1, order).

For "Order" sort the vector and returns the indices of each element that it has before the sorting. The result is the same as order(x) but for the same exactly results set argument "stable" to "TRUE".

Author(s)

Manos Papadakis

R implementation and documentation: Manos Papadakis <papadakm95@gmail.com>.

See Also

colsums, coldiffs, colMedians, colprods
Column and row-wise products

Examples

```r
x <- matrix( runif(10 * 10), ncol = 10 )
res< - colOrder(x)
res<-apply(x, 2, order)
res<-rowOrder(x)
t(apply(x, 1, order))

y <- rnorm(100)
b <- Order(y)
a <- order(y)
all.equal(a,b) ## false because it is not stable
b <- Order(y,stable=TRUE)
all.equal(a,b) ## true because it is stable

x<-y<-b<-a<-NULL
```

Column and row-wise products

Description

Column and row-wise products.

Usage

```r
colprods(x, method = "direct")
rowprods(x)
```

Arguments

- **x**: A matrix with numbers.
- **method**: The type of colCumProds to use. For direct multiplication use "direct" or "expsumlog" for a more numerically stable, but slower way.

Details

The product of the numbers in a matrix is returned either column-wise or row-wise.

Value

A vector with the column or the row products.

Author(s)

Manos Papadakis
R implementation and documentation: Manos Papadakis <papadas95@gmail.com>.
Column and row-wise range of values of a matrix

See Also
colsums, coldiffs, colMedians

Examples

```r
x <- matrix( runif(100 * 10), ncol = 10 )
res <- colprods(x)
res <- rowprods(x)
x <- NULL
```

Description

Column and row-wise range of values of a matrix.

Usage

```r
colrange(x, cont = TRUE)
rowrange(x, cont = TRUE)
```

Arguments

- `x` A numerical matrix with data.
- `cont` If the data are continuous, leave this TRUE and it will return the range of values for each variable (column). If the data are integers, categorical, or if you want to find out the number of unique numbers in each column set this to FALSE.

Value

A vector with the relevant values.

Author(s)

Manos Papadakis

R implementation and documentation: Manos Papadakis <papadakm95@gmail.com>.

See Also

colMins, colMaxs, rowMins, rowMaxs, nth, colMedians, colVars, colSort, rowSort
Column and row-wise ranks

Examples

```r
x <- matrix( rnorm(100 * 100), ncol = 100 )

a1 <- colrange(x)
a2 <- apply(x, 2, function(x) diff( range(x) )
all.equal(a1, a2)

a1 <- rowrange(x)
a2 <- apply(x, 1, function(x) diff( range(x) )
all.equal(a1, a2)
```

x<-a1<-a2<-NULL

---

Column and row-wise ranks

Column and row-wise ranks

Description

Column and row-wise ranks.

Usage

```r
colRanks(x, method = "average", descending = FALSE, stable = FALSE, parallel = FALSE)
rowRanks(x, method = "average", descending = FALSE, stable = FALSE, parallel = FALSE)
```

Arguments

- `x`: A numerical matrix with the data.
- `parallel`: A boolean value for parallel version.
- `method`: A character string for choosing method. Must be one of "average", "min", "max", "first".
- `descending`: A boolean value (TRUE/FALSE) for sorting the vector in descending order. By default sorts the vector in ascending.
- `stable`: A boolean value (TRUE/FALSE) for choosing a stable sort algorithm. Stable means that discriminates on the same elements. Only for the method "first".

Details

For each column or row of a matrix the ranks are calculated and they are returned. The initial matrix is gone.

Value

A matrix with the column or row-wise ranks.
Column and row-wise Shuffle

Description

Column and row-wise shuffle of a matrix.

Usage

colShuffle(x)
rowShuffle(x)

Arguments

x                A matrix with the data.

Details

The functions is written in C++ in order to be as fast as possible.

Value

A vector with the column/row Shuffle.

Author(s)

R implementation and documentation: Manos Papadakis <papadakm95@gmail.com>.

Examples

x <- matrnorm(100, 10)
a1 <- colRanks(x)
a2 <- apply(x, 2, rank)
b1 <- rowRanks(x)
b2 <- apply(x, 1, rank)

x<-a1<-a2<-b1<-b2<-NULL
See Also

Median, colVars, colMeans (built-in R function)

Examples

```r
x <- matrix( rnorm(100 * 100), ncol = 100 )
system.time( colShuffle(x) )
system.time( rowShuffle(x) )
```

```r
x<-NULL
```

Description

Column and row-wise sums of a matrix.

Usage

```r
colsums(x, indices = NULL, parallel = FALSE, na.rm = FALSE)
```

```r
rowsums(x, indices = NULL, parallel = FALSE, na.rm = FALSE)
```

Arguments

- `x`: A numerical matrix with data.
- `indices`: An integer vector with the indices to sum the columns/rows.
- `parallel`: Do you want to do it in parallel in C++? TRUE or FALSE. Does't work with argument "indices".
- `na.rm`: A logical value indicating to remove NAs. The algorithm run in parallel so do not use with option parallel.

Value

A vector with sums.

Author(s)

Manos Papadakis
R implementation and documentation: Manos Papadakis <papadakm95@gmail.com>.

See Also

colMedians, colmeans, colVars
Examples

```r
x <- matrix(rpois(500 * 100, 10), ncol = 100)
x1 <- colsums(x)
x2 <- colSums(x)
all.equal(x1, x2)

x1 <- rowsums(x)
x2 <- rowSums(x)
all.equal(x1, x2)
```

```
x <- x1 <- x2 <- NULL
```

Description

Column and row-wise tabulate of a matrix.

Usage

```r
colTabulate(x, max_number = max(x))
rowTabulate(x, max_number = max(x))
```

Arguments

- `x`: An integer matrix with the data. The numbers must start from 1, i.e. 1, 2, 3, 4,... No zeros are allowed. Anything else may cause a crash.
- `max_number`: The maximum value of vector `x`. If you know which is the max number use this argument for faster results or by default `max(x)`.

Details

The functions is written in C++ in order to be as fast as possible.

Value

A matrix where in each column the command "tabulate" has been performed. The number of rows of the returned matrix will be equal to the `max_number` if given. Otherwise, the functions will find this number.

Author(s)

R implementation and documentation: Manos Papadakis <papadakm95@gmail.com>.

See Also

`colShuffle`, `colVars`, `colmeans`
Column and row-wise variances and standard deviations

Examples

```r
x <- matrix( rbinom(100 * 100, 4, 0.5), ncol = 100 )
system.time( colTabulate(x) )
x <- t(x)
system.time( rowTabulate(x) )
```

x<-NULL

---

Column and row-wise variances and standard deviations

Column and row-wise variances and standard deviations of a matrix

Description

Column and row-wise variances and standard deviations of a matrix

Usage

```r
colVars(x, suma = NULL, std = FALSE, na.rm = FALSE, parallel = FALSE)
rowVars(x, suma = NULL, std = FALSE, na.rm = FALSE, parallel = FALSE)
```

Arguments

- `x`: A matrix with the data.
- `suma`: If you already have the column sums vector supply it, otherwise leave it NULL. Deprecated.
- `std`: A boolean variable specifying whether you want the variances (FALSE) or the standard deviations (TRUE) of each column.
- `na.rm`: TRUE or FALSE for remove NAs if exists.
- `parallel`: Should parallel implementations take place in C++? The default value is FALSE.

Details

We found this on stackoverflow which was created by David Arenburg. We then modified the function to match the sums type formula of the variance, which is faster.

Value

A vector with the column variances or standard deviations.

Author(s)

Michail Tsagris

R implementation and documentation: Michail Tsagris <mtsagris@uoc.gr> and Manos Papadakis <papadakm95@gmail.com>. 
Column and rows-wise mean absolute deviations

See Also

  colmeans, colMedians, colrange

Examples

  x <- matrix( rnorm(100 * 100), ncol = 100 )
  a2 <- colVars(x)
  x<-a2<-NULL

Description

  Column and row-wise mean absolute deviations.

Usage

  colMads(x, method = "median", na.rm = FALSE, parallel = FALSE)
  rowMads(x, method = "median", na.rm = FALSE, parallel = FALSE)

Arguments

  x A matrix with the data.
  method A character vector with values "median", for median absolute deviation or "mean", for mean absolute deviation.
  na.rm A logical value TRUE/FALSE to remove NAs.
  parallel A boolean value for parallel version.

Details

  The functions is written in C++ in order to be as fast as possible.

Value

  A vector with the column-wise mean absolute deviations.

Author(s)

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

See Also

  colMedians, rowMedians, colVars, colmeans, colMeans (built-in R function)
Examples

```r
x <- matrix( rnorm(100 * 100), ncol = 100 )
system.time( a <- colMads(x) )

x<-NULL
```

Description

Column-row wise minima and maxima of two matrices.

Usage

```r
colPmax(x, y)
colPmin(x, y)
```

Arguments

- `x`: A numerical vector with numbers.
- `y`: A numerical vector with numbers.

Details

The parallel minima or maxima are returned. This are the same as the base functions pmax and pmin.

Value

A numerical vector/matrix with numbers, whose length is equal to the length of the initial matrices containing the maximum or minimum between each pair.

Author(s)

Manos Papadakis

R implementation and documentation: Manos Papadakis <papadakm95@gmail.com>.

See Also

Sort, colMins, colMaxs, colMedians
Column-wise differences

Examples

```r
x <- matrix(rnorm(100),10,10)
y <- matrix(rnorm(100),10,10)
res<-colPmax(x, y)
res<-colPmin(x, y)
x<-y<-NULL
```

Description

Column-wise differences.

Usage

coldiffs(x)

Arguments

x

A matrix with numbers.

Details

This function simply does this function x[, -1] - x[, -k], where k is the last column of the matrix x. But it does it a lot faster. That is, 2nd column - 1st column, 3rd column - 2nd column, and so on.

Value

A matrix with one column less containing the differences between the successive columns.

Author(s)

Manos Papadakis

R implementation and documentation: Manos Papadakis <papadakm95@gmail.com>.

See Also

dist, dista, colmeans

Examples

```r
x <- matrix( rnorm(50 * 10), ncol = 10 )
res<-coldiffs(x)
x<-NULL
```
Column-wise kurtosis and skewness coefficients

Description

Column-wise kurtosis and skewness coefficients.

Usage

colkurtosis(x, pvalue = FALSE)
colskewness(x, pvalue = FALSE)

Arguments

x
A matrix with the data, where the rows denote the samples and the columns are the variables.
pvalue
If you want a hypothesis test that the skewness or kurtosis are significant set this to TRUE. This checks whether the skewness is significantly different from 0 and whether the kurtosis is significantly different from 3.

Details

The skewness and kurtosis coefficients are calculated. For the skewness coefficient we use the sample unbiased version of the standard deviation. For the kurtosis, we do not subtract 3.

Value

If "pvalue" is FALSE, a vector with the relevant coefficient. Otherwise a matrix with two columns. The kurtosis or skewness coefficient and the p-value from the hypothesis test that they are significantly different from 3 or 0 respectively.

Author(s)

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

See Also

skew, skew.test2, colMedians, colmeans, colVars, sftests
### Examples

```r
## 200 variables, hence 200 F-tests will be performed
x = matrix( rnorm(200 * 50), ncol = 50 )
## 200 observations in total
system.time( colkurtosis(x) )
system.time( colskewness(x) )
x <- NULL
```

### Description

Column-wise matching coefficients.

### Usage

```r
match.coefs(x, y = NULL, ina, type = "jacc")
```

### Arguments

- **x**: A matrix with the data, where the rows denote the samples and the columns are the variables.
- **y**: A second matrix with the data of the second group. If this is NULL (default value) then the argument `ina` must be supplied. Notice that when you supply the two matrices the procedure is two times faster.
- **ina**: A numerical vector with 1s and 2s indicating the two groups. Be careful, the function is designed to accept only these two numbers. In addition, if your "y" is NULL, you must specify "ina".
- **type**: This denotes the type of matching coefficient to calculate. For the Jaccard index put "jacc". For the simple matching coefficient put "smc" or else both of them will be calculated.

### Details

Two matrices are given as input and for each column matching coefficients are calculated, either the Jaccard or the simple matching coefficient or both.

### Value

A matrix with one or two columns, depending on the type you have specified. If you specify "both", there will be two columns, if you specify "jacc" or "smc" then just one column.
Column-wise minimum and maximum

Description

Column-wise minimum and maximum of a matrix.

Usage

```r
colMins(x, value = FALSE, parallel = FALSE)
```

```r
colMaxs(x, value = FALSE, parallel = FALSE)
```

```r
colMinsMaxs(x)
```

Arguments

- `x`: A numerical matrix with data.
- `value`: If the value is FALSE it returns the indices of the minimum/maximum, otherwise it returns the minimum and maximum values.
- `parallel`: Do you want to do it in parallel in C++? TRUE or FALSE. The parallel will return the minimum/maximum value only. It will never return the indices.

Value

A vector with the relevant values.

See Also

`odds`, `colTabulate`

Examples

```r
x <- matrix(rbinom(400 * 10, 1, 0.5), ncol = 10)
y <- matrix(rbinom(400 * 10, 1, 0.5), ncol = 10)
a <- match.coefs(x, y, type = "both")
x <- NULL
y <- NULL
```
Column-wise MLE of some univariate distributions

Author(s)
Manos Papadakis
R implementation and documentation: Manos Papadakis <papadakm95@gmail.com>.

See Also
rowMins, rowMaxs, nth, colrange, colMedians, colVars, colSort, rowSort

Examples

```r
x <- matrix(rnorm(100 * 200), ncol = 200)

s1 <- colMins(x)
s2 <- apply(x, 2, min)

s1 <- colMaxs(x)
s2 <- apply(x, 2, max)

s1 <- colMinsMaxs(x)
s2 <- c(apply(x, 2, min), apply(x, 2, max))
x <- s1 <- s2 <- NULL
```

Description

Column-wise MLE of some univariate distributions.

Usage

colexpmle(x)
colexp2.mle(x)
colgammmamle(x, tol = 1e-07)
colinvgauss.mle(x)
collaplace.mle(x)
collindley.mle(x)
colmaxboltz.mle(x)
colnormal.mle(x)
colpareto.mle(x)
colrayleigh.mle(x)
colvm.mle(x, tol = 1e-07)
colweibull.mle(x, tol = 1e-09, maxiters = 100, parallel = FALSE)
colnormlog.mle(x)

Column-wise MLE of some univariate distributions

Arguments

x
A numerical matrix with data. Each column refers to a different vector of observations of the same distribution. For exponential, 2 parameter exponential, Weibull, gamma, inverse Gaussian, Maxwell-Boltzman, Lindley, Rayleigh and Pareto distributions, the numbers must be greater than zero. For the Poisson and geometric distributions, the numbers must be integers, 0, 1, 2,... For the Normal and Laplace distribution the numbers can take any value. The von Mises distribution takes values between 0 and 2 * pi (radians).

tol
The tolerance value to terminate the Newton-Fisher algorithm.

maxiters
The maximum number of iterations to implement.

parallel
Do you want to calculations to take place in parallel? The default value is FALSE

Details

For each column, the same distribution is fitted and its parameter and log-likelihood are computed.

Value

A matrix with two, three or five (for the colnormlog.mle) columns. The first one or the first two contain the parameter(s) of the distribution and the other columns contain the log-likelihood values.

Author(s)

Michail Tsagris and Stefanos Fafalios

R implementation and documentation: Michail Tsagris <mtsagris@uoc.gr> and Stefanos Fafalios <stefanosfafalios@gmail.com>

References


See Also

vm.mle,poisson.mle,normal.mle,gammamle
**Examples**

```r
x <- matrix(rnorm(1000 * 50), ncol = 50)
a <- colnormal.mle(x)
b <- collaplace.mle(x)
x <- NULL
```

**Description**

Column-wise true/false value of a matrix.

**Usage**

```r
colTrue(x)
colFalse(x)
colTrueFalse(x)
```

**Arguments**

`x`  
A logical matrix with data.

**Value**

An integer vector where item "i" is the number of the true/false values of "i" column.

**Author(s)**

Manos Papadakis  
R implementation and documentation: Manos Papadakis <papadakm95@gmail.com>.

**See Also**

`rowMins, rowFalse, nth, colrange, colMedians, colVars, colSort, rowSort, rowTrue`

**Examples**

```r
x <- matrix(as.logical(rbinom(100*100,1,.5)),100,100)
s1 <- colTrue(x)
s1 <- colFalse(x)
s1 <- colTrueFalse(x)
x<-s1<-NULL
```
Description

Column-wise uniformity tests for circular data.

Usage

colwatsons(u)

Arguments

u A numeric matrix containing the circular data which are expressed in radians. Each column is a different sample.

Details

These tests are used to test the hypothesis that the data come from a circular uniform distribution. The Kuiper test is much more time consuming and this is why it not implemented yet. Once we figure out a way to make it fast, we will include it.

Value

A matrix with two columns, the value of the test statistic and its associated p-value.

Author(s)

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

References


See Also

watson, vmf.mle, rvonmises

Examples

x <- matrix( rvonmises(n = 50 * 10, m = 2, k = 0), ncol = 10 )
res <- colwatsons(x)
x <- NULL
Column-wise Yule’s Y (coefficient of colligation)

Description

Column-wise Yule’s Y (coefficient of colligation).

Usage

col.yule(x, y = NULL, ina)

Arguments

x A matrix with 0 and 1. Every column refers to a different sample or variable.
y A second matrix, of the same dimensions as x, with 0 and 1. Every column refers to a different sample or variable.
ina If y is NULL, ina must be specified. This is a numeric vector with 1s and 2s, indicating the group of each row.

Details

Yule’s coefficient of colligation is calculated for every column.

Value

A vector with Yule’s Y, one for every column of x is returned.

Author(s)

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

References


See Also

yule, odds

Examples

x <- matrix(rbinom(300 * 10, 1, 0.5), ncol = 10)
ina <- rep(1:2, each = 150)
res <- col.yule(x, ina = ina)
Convert a dataframe to matrix

Description

Convert a dataframe to matrix.

Usage

data.frame.to_matrix(x, col.names = NULL, row.names = NULL)

Arguments

x      A Numeric matrix with data and NAs.
col.names A boolean value for keeping the colnames for argument x or a character vector for the new colnames.
row.names A boolean value for keeping the rownames for argument x or a character vector for the new rownames.

Details

This function converts a dataframe to matrix. Even if there are factors, the function converts them into numerical values. Attributes are not allowed for now.

Value

A matrix which has the numerical values from the dataframe.

Author(s)

Manos Papadakis

R implementation and documentation: Manos Papadakis <papadakm95@gmail.com>

See Also

Match, is.symmetric, permutation

Examples

res <- data.frame.to_matrix(iris)
Convert R function to the Rfast’s corresponding

Description

Convert R function to the Rfast’s corresponding.

Usage

as.Rfast.function(Rfunction.name, margin=NULL)

Arguments

- Rfunction.name: An character value with the name of the function.
- margin: A logical function for return the column-row wise function.

Details

Given the name of R function, it returns the corresponding function’s name from Rfast.

Value

The corresponding Rfast function.

Author(s)

Manos Papadakis and Michail Tsagris

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

See Also

colsums, colMedians, colVars

Examples

res<-as.Rfast.function("var")
Correlation based forward regression

Description
Correlation based forward regression.

Usage
cor.fsreg(y, x, ystand = TRUE, xstand = TRUE, threshold = 0.05, tolb = 2, tolr = 0.02, stopping = "BIC")

Arguments
y
A numerical vector.
x
A matrix with data, the predictor variables.
ystand
If this is TRUE the response variable is centered. The mean is subtracted from every value.
xstand
If this is TRUE the independent variables are standardised.
threshold
The significance level, set to 0.05 by default. Bear in mind that the logarithm of it is used, as the logarithm of the p-values is calculated at every point. This will avoid numerical overflows and small p-values, less than the machine epsilon, being returned as zero.
tolb
If we see only the significance of the variables, many may enter the linear regression model. For this reason, we also use the BIC as a way to validate the inclusion of a candidate variable. If the BIC difference between two successive models is less than the tolerance value, the variable will not enter the model, even if it statistically significant. Set it to 0 if you do not want this extra check.
tolr
This is an alternative to the BIC change and it uses the adjusted coefficient of determination. If the increase in the adjusted $R^2$ is more than the tolr continue.

stopping
This refers to the type of extra checking to do. If you want the BIC check, set it to "BIC". If you want the adjusted $R^2$ check set this to "ar2". Or, if you want both of them to take place, both of these criteria to be satisfied make this "BICR2".

Details
The forward regression tries one by one the variables using the F-test, basically partial F-test every time for the latest variable. This is the same as testing the significance of the coefficient of this latest entered variable. Alternatively the correlation can be used and this case the partial correlation coefficient. There is a direct relationship between the t-test statistic and the partial correlation coefficient. Now, instead of having to calculate the test statistic, we calculate the partial correlation coefficient. Using Fisher’s z-transform we get the variance immediately. The partial correlation coefficient, using Fisher’s z-transform, and the partial F-test (or the coefficient’s t-test statistic) are not identical. They will be identical for large sample sizes though.
Value
A matrix with three columns, the index of the selected variables, the logged p-value and the test statistic value and the BIC or adjusted $R^2$ of each model. In the case of stopping="BICR2" both of these criteria will be returned.

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

References

See Also
score.glms, univglms, logistic_only, poisson_only, regression

Examples
```r
## 200 variables, hence 200 univariate regressions are to be fitted
x <- matrnorm(200, 100)
y <- rnorm(200)
system.time( cor.fsreg(y, x) )
x <- NULL
```

Description
Correlations between pairs of variables.

Usage
corpairs(x, y, rho = NULL, logged = FALSE, parallel = FALSE)

Arguments
- `x`: A matrix with real valued data.
- `y`: A matrix with real valued data whose dimensions match those of `x`.
- `rho`: This can be a vector of assumed correlations (equal to the number of variables or the columns of `x` or `y`) to be tested. If this is not the case, leave it NULL and only the correlations will be returned.
- `logged`: Should the p-values be returned (FALSE) or their logarithm (TRUE)? This is taken into account only if "rho" is a vector.
- `parallel`: Should parallel implementations take place in C++? The default value is FALSE.
Details

The paired correlations are calculated. For each column of the matrices x and y the correlation between them is calculated.

Value

A vector of correlations in the case of "rho" being NULL, or a matrix with two extra columns, the test statistic and the (logged) p-value.

Author(s)

Michail Tsagris

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

References


See Also

correls, allbetas, mvbetas

Examples

x <- matrnorm(100, 100)
y <- matrnorm(100, 100)

system.time( corpairs(x, y) )
a <- corpairs(x, y)
x <- corpairs(x, y)
x <- NULL
y <- NULL
Correlations

**Description**

Correlation between a vector and a set of variables.

**Usage**

```r
correls(y, x, type = "pearson", a = 0.05, rho = 0)
groupcorrels(y, x, type = "pearson", ina)
```

**Arguments**

- `y`: A numerical vector.
- `x`: A matrix with the data.
- `type`: The type of correlation you want. "pearson" and "spearman" are the two supported types for the "correls" because their standard error is easily calculated. For the "groupcorrels" you can also put "kendall" because no hypothesis test is performed in that function.
- `a`: The significance level used for the confidence intervals.
- `rho`: The value of the hypothesised correlation to be used in the hypothesis testing.
- `ina`: A factor variable or a numeric variable indicating the group of each observation.

**Details**

The functions use the built-in function "cor" which is very fast and then includes confidence intervals and produces a p-value for the hypothesis test.

**Value**

For the "correls" a matrix with 5 columns; the correlation, the p-value for the hypothesis test that each of them is equal to "rho", the test statistic and the $a/2\%$ lower and upper confidence limits. For the "groupcorrels" a matrix with rows equal to the number of groups and columns equal to the number of columns of `x`. The matrix contains the correlations only, no statistical hypothesis test is performed.

**Author(s)**

Michail Tsagris

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

**See Also**

allbetas, univglm
Examples

```r
x <- matrnorm(60, 100)
y <- rnorm(60)
r <- cor(y, x)  ## correlation of y with each of the xs
a <- allbetas(y, x)  ## # the coefficients of each simple linear regression of y with x
b <- correls(y, x)
ina <- rep(1:2, each = 30)
b2 <- groupcorrels(y, x, ina = ina)
x <- NULL
```

Covariance and correlation matrix

`cova(x, center = FALSE, large = FALSE)`

`cora(x, large = FALSE)`

Arguments

- `x`: A matrix with data. It has to be matrix, if it is data.frame for example the function does not turn it into a matrix.
- `center`: If you want to center the data prior to applying the cross product of the mateix set this equal to TRUE, otherwise leave it NULL.
- `large`: If you have large matrices, with thousands of rows and or many tens or hundreds of columns set this equal to TRUE in order to use Rfast's `Crossprod` or `Tcrossprod` functions. These functions are twice or up to 3 times faster than the corresponding built-in functions.

Details

The calculations take place faster than the built-in functions `cor` as the number of variables increases. This is true if the number of variables is high, say from 500 and above. The "cova" on the other hand is always faster. For the "cova" in specific, we have an option to center the data prior to the cross product. This can be more stable if you have many tens of thousands of rows due to numerical issues that can arise.

For the correlation matrix we took the code from here

Value

The covariance or the correlation matrix.

Author(s)

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

See Also

colVars, cor, cov

Examples

```r
x <- matrnorm(100, 40)
s1 <- cov(x)
s2 <- cova(x)
all.equal(s1, s2)
x <- NULL
```
Cross-Validation for the k-NN algorithm

**Value**

For the `cox.poisrat` a vector with three elements, the ratio and the lower and upper confidence interval limits. For the `col.coxpoisrat` a matrix with three columns, the ratio and the lower and upper confidence interval limits.

**Author(s)**

Michail Tsagris

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

**References**


**See Also**

correls, Table

**Examples**

```r
x <- rpois(100, 10)
y <- rpois(100, 10)
res <- cox.poisrat(x, y)
```

---

Cross-Validation for the k-NN algorithm

**Description**

Cross-Validation for the k-NN algorithm.

**Usage**

```r
knn.cv(folds = NULL, nfolds = 10, stratified = FALSE, seed = FALSE, y, x, k, dist.type = "euclidean", type = "C", method = "average", freq.option = 0, pred.ret = FALSE, mem.eff = FALSE)
```

**Arguments**

- `folds` A list with the indices of the folds.
- `nfolds` The number of folds to be used. This is taken into consideration only if "folds" is NULL.
stratified  Do you want the folds to be selected using stratified random sampling? This preserves the analogy of the samples of each group. Make this TRUE if you wish, but only for the classification. If you have regression (type = "R"), do not put this to TRUE as it will cause problems or return wrong results.

seed     If you set this to TRUE, the same folds will be created every time.

y        A vector of data. The response variable, which can be either continuous or categorical (factor is acceptable).

x        A matrix with the available data, the predictor variables.

k        A vector with the possible numbers of nearest neighbours to be considered.

dist.type The type of distance to be used, "euclidean" or "manhattan".

type     Do you want to do classification ("C") or regression ("R")?

method   If you do regression (type = "R"), then how should the predicted values be calculated? Choose among the average ("average"), median ("median") or the harmonic mean ("harmonic") of the closest neighbours.

dist.type The type of distance to be used, "euclidean" or "manhattan".

freq.option If classification (type = "C") and ties occur in the prediction, more than one class have the same number of k nearest neighbours, there are three strategies available. Option 0 selects the first most frequent encountered. Option 1 randomly selects the most frequent value, in the case that there are duplicates.

pred.ret  If you want the predicted values returned set this to TRUE.

mem.eff  Boolean value indicating a conservative or not use of memory. Lower usage of memory/Having this option on will lead to a slight decrease in execution speed and should ideally be on when the amount of memory in demand might be a concern.

Details

The concept behind k-NN is simple. Suppose we have a matrix with predictor variables and a vector with the response variable (numerical or categorical). When a new vector with observations (predictor variables) is available, its corresponding response value, numerical or categorical, is to be predicted. Instead of using a model, parametric or not, one can use this ad hoc algorithm.

The k smallest distances between the new predictor variables and the existing ones are calculated. In the case of regression, the average, median, or harmonic mean of the corresponding response values of these closest predictor values are calculated. In the case of classification, i.e. categorical response value, a voting rule is applied. The most frequent group (response value) is where the new observation is to be allocated.

This function does the cross-validation procedure to select the optimal k, the optimal number of nearest neighbours. The optimal in terms of some accuracy metric. For the classification it is the percentage of correct classification and for the regression the mean squared error.

Value

A list including:

preds     If pred.ret is TRUE the predicted values for each fold are returned as elements in a list.

crit      A vector whose length is equal to the number of k and is the accuracy metric for each k.
Cross-Validation for the k-NN algorithm using the arc cosinus distance

Author(s)
Marios Dimitriadis
R implementation and documentation: Marios Dimitriadis <kmdimitriadis@gmail.com>

References

See Also
dirknn.cv

Examples
x <- as.matrix(iris[, 1:4])
y <- iris[, 5]
mod <- knn.cv(folds = NULL, nfolds = 10, stratified = TRUE, seed = FALSE, y = y, x = x, k = c(3, 4), dist.type = "euclidean", type = "C", method = "average", freq.option = 0, pred.ret = FALSE, mem.eff = FALSE)

Description
Cross-Validation for the k-NN algorithm using the arc cosinus distance.

Usage
dirknn.cv(y, x, k = 5:10, type = "C", folds = NULL, nfolds = 10, stratified = TRUE, seed = FALSE, parallel = FALSE, pred.ret = FALSE)

Arguments
y A vector of data. The response variable, which can be either continuous or categorical (factor is acceptable).
x A matrix with the available data, the predictor variables.
k A vector with the possible numbers of nearest neighbours to be considered.
**Cross-Validation for the k-NN algorithm using the arc cosinus distance**

- **type**: If your response variable y is numerical data, then this should be "R" (regression) or "WR" for distance weighted based nearest neighbours. If y is in general categorical set this argument to "C" (classification) or to "WC" for distance weighted based nearest neighbours.
- **folds**: A list with the indices of the folds.
- **nfolds**: The number of folds to be used. This is taken into consideration only if "folds" is NULL.
- **stratified**: Do you want the folds to be selected using stratified random sampling? This preserves the analogy of the samples of each group. Make this TRUE if you wish, but only for the classification. If you have regression (type = "R"), do not put this to TRUE as it will cause problems or return wrong results.
- **seed**: If you set this to TRUE, the same folds will be created every time.
- **parallel**: Do you want the calculations to take place in parallel? The default value is FALSE.
- **pred.ret**: If you want the predicted values returned set this to TRUE.

**Details**

The concept behind k-NN is simple. Suppose we have a matrix with predictor variables and a vector with the response variable (numerical or categorical). When a new vector with observations (predictor variables) is available, its corresponding response value, numerical or categorical, is to be predicted. Instead of using a model, parametric or not, one can use this ad hoc algorithm.

The k smallest distances between the new predictor variables and the existing ones are calculated. In the case of regression, the average, median, or harmonic mean of the corresponding response values of these closest predictor values are calculated. In the case of classification, i.e. categorical response value, a voting rule is applied. The most frequent group (response value) is where the new observation is to be allocated.

This function does the cross-validation procedure to select the optimal k, the optimal number of nearest neighbours. The optimal in terms of some accuracy metric. For the classification it is the percentage of correct classification and for the regression the mean squared error.

**Value**

A list including:

- **preds**: If pred.ret is TRUE the predicted values for each fold are returned as elements in a list.
- **crit**: A vector whose length is equal to the number of k and is the accuracy metric for each k. For the classification case it is the percentage of correct classification. For the regression case the mean square of prediction error.

**Author(s)**

Michail Tsagris

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

References

See Also
dirknn,knn.cv,knn

Examples
x <- as.matrix(iris[, 1:4])
x <- x / sqrt( Rfast::rowsums(x^2) )
y <- iris[, 5]
mod <- dirknn.cv(y = y, x = x, k = c(3, 4) )

Description
Deep copy.

Usage
env.copy(x, all.names=FALSE)

Arguments
x An environment object.
all.names An logical value (TRUE or FALSE). Copy all the hidden variables or not.

Details
Deep copy of the environment object.

Value
A copy of the first argument.

Author(s)
R implementation and documentation: Manos Papadakis <papadakm95@gmail.com>.

See Also
colShuffle, colVars, colmeans, read.directory
Examples

x <- new.env()
x$imaginary <- NULL
x$real <- NULL

# you can library the package and just press x and R will understand
# and search automatically for a function to print the environment
x

y <- env.copy(x)

x$real <- 10

x$real == y$real # FALSE

Density of the multivariate normal and t distributions

Description

Density of the multivariate normal and t distributions.

Usage

\[
\text{dmvnorm}(x, \mu, \sigma, \text{logged} = \text{FALSE})
\]

\[
\text{dmvt}(x, \mu, \sigma, \nu, \text{logged} = \text{FALSE})
\]

Arguments

- **x**: A numerical matrix with the data. The rows correspond to observations and the columns to variables.
- **mu**: The mean vector.
- **sigma**: The covariance matrix.
- **nu**: The degrees of freedom for the multivariate t distribution.
- **logged**: Should the logarithm of the density be returned (TRUE) or not (FALSE)?

Details

The (log) density of the multivariate normal distribution is calculated for given mean vector and covariance matrix.

Value

A numerical vector with the density values calculated at each vector (row of the matrix x).
Design Matrix

Author(s)

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

References


See Also

rmvnorm, rmvt, mvnorm.mle, iag.mle

Examples

```r
x <- matrnorm(100, 20)
mu <- colmeans(x)
s <- cova(x)
a1 <- dmvnorm(x, mu, s)
a2 <- dmvt(x, mu, s, 1)
x <- NULL
```

Design Matrix

Design Matrix

Description

Design Matrix.

Usage

design_matrix(x, ones = TRUE)

Arguments

x  A character vector or a factor type vector or a dataframe. Do not supply a numerical vector.
ones  A boolean variable specifying whether to include the ones in the design matrix or not. The default value is TRUE.

Details

This function implements the R’s "model.matrix" function and is used only when the x is a factor/charactervector or Dataframe.
Diagonal Matrix

Value
Returns the same matrix with model.matrix.

Author(s)
Manos Papadakis
R implementation and documentation: Manos Papadakis <papadakm95@gmail.com>

See Also
model.matrix

Examples
a <- design_matrix( iris[, 5] )
b <- model.matrix( ~ iris[,5] ) ## R's built-in function
all.equal(as.vector(a),as.vector(b)) ## true
a<-b<-NULL

Description
Fill the diagonal of a matrix or create a diagonal and initialize it with a specific value.

Usage
Diag.fill(x,v=0)
Diag.matrix(len,v=0)

Arguments
x A matrix with data.
len Number of columns or rows.
v Value or vector to initialize the diagonal of a matrix. By default "v=0".

Value
Diag.fill returns a diagonal matrix where all the elements in the diagonal are equal to "v".
Diag.matrix returns a diagonal matrix where has dimension "len,len" and all the elements in the diagonal are equal to "v". It is fast for huge matrices with dimensions more than [row,col] = [500,500]

Author(s)
Manos Papadakis
R implementation and documentation: Manos Papadakis <papadakm95@gmail.com>.
Distance between vectors and a matrix

Description
Distance between vectors and a matrix.

Usage

dista(xnew, x, type = "euclidean", k = 0, index = FALSE, trans = TRUE, square = FALSE)

Arguments

xnew         A matrix with some data or a vector.
x            A matrix with the data, where rows denotes observations (vectors) and the columns contain the variables.
type         This can be either "euclidean" or "manhattan".
k            Should the k smaller distances or their indices be returned? If k > 0 this will happen.
index        In case k is greater than 0, you have the option to get the indices of the k smallest distances.
trans        Do you want the returned matrix to be transposed? TRUE or FALSE.
square       If you choose "euclidean" as the method, then you can have the option to return the squared Euclidean distances by setting this argument to TRUE.

Details
The target of this function is to calculate the distances between xnew and x without having to calculate the whole distance matrix of xnew and x. The latter does extra calculations, which can be avoided.
Distance correlation

Description

Distance correlation.

Usage

dcor(x, y)
bcdcor(x, y)

Arguments

x A numerical matrix.
y A numerical matrix.
Details

The distance correlation or the bias corrected distance correlation of two matrices is calculated. The latter one is used for the hypothesis test that the distance correlation is zero (see \texttt{dcor.ttest}).

Value

The value of the distance correlation of the bias corrected distance correlation.

Author(s)

Manos Papadakis

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

References


See Also

\texttt{dcov, dcor.ttest, edist}

Examples

```r
x <- as.matrix(iris[1:50, 1:4])
y <- as.matrix(iris[51:100, 1:4])
res <- dcor(x, y)
res <- bcdcor(x, y)
x <- y <- NULL
```

Description

Distance matrix.

Usage

```r
Dist(x, method = "euclidean", square = FALSE, p = 0, vector = FALSE)
vecdist(x)
```
Distance matrix

Arguments

A matrix with data. The distances will be calculated between pairs of rows. In the case of `vecdist` this is a vector. For the haversine distance it must be a matrix with two columns, the first column is the latitude and the second the longitude.

`method` This is either "euclidean", "manhattan", "canberra1", "canberra2", "minimum", "maximum", "minkowski", "bhattacharyya", "hellinger", "kullback_leibler", "jensen_shannon" or "haversine". The last two options are basically the same.

`square` If you choose "euclidean" or "hellinger" as the method, then you can have the option to return the squared Euclidean distances by setting this argument to TRUE.

`p` This is for the the Minkowski, the power of the metric.

`vector` For return a vector instead a matrix.

Details

The distance matrix is computer with an extra argument for the Euclidean distances. The "kullback_leibler" refers to the symmetric Kullback-Leibler divergence.

Value

A square matrix with the pairwise distances.

Author(s)

Manos Papadakis.

R implementation and documentation: Manos Papadakis <papadakm95@gmail.com>.

References


See Also

dista, colMedians

Examples

```r
x <- matrix(rnorm(50 * 10), ncol = 10)
a1 <- Dist(x)
a2 <- as.matrix(dist(x))
x <- a1 <- a2 <- NULL
```
Distance variance and covariance

Description

Distance variance and covariances.

Usage

dvar(x)
dcov(x, y)

Arguments

x
A numerical matrix or a vector.
y
A numerical matrix or a vector.

Details

The distance variance of a matrix/vector or the distance covariance of two matrices is calculated. For the distance variance of a vector we use the fast method of Huo and Szekely (2016).

Value

The distance covariance or distance variance.

Author(s)

Manos Papadakis

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

References


See Also

dcor, edist
Examples

```r
x <- as.matrix(iris[1:50, 1:4])
y <- as.matrix(iris[51:100, 1:4])
res <- dcov(x, y)
res <- dvar(x[, 1])
```

Description

Eigenvalues in high dimensional (n«p) principal component analysis.

Usage

```r
hd.eigen(x, center = TRUE, scale = FALSE, k = NULL, vectors = FALSE, large = FALSE)
```

Arguments

- `x`: A numerical $n \times p$ matrix with data where the rows are the observations and the columns are the variables.
- `center`: Do you want your data centered? TRUE or FALSE.
- `scale`: Do you want each of your variables scaled, i.e. to have unit variance? TRUE or FALSE.
- `k`: If you want a specific number of eigenvalues and eigenvectors set it here, otherwise all eigenvalues (and eigenvectors if requested) will be returned.
- `vectors`: Do you want the eigenvectors be returned? By default this is FALSE.
- `large`: If you have large matrices, with thousands of rows and or many tens or hundreds of columns set this equal to TRUE in order to use Rfast’s `Crossprod` or `Tcrossprod` functions. These functions are twice or up to 3 times faster than the corresponding built-in functions.

Details

When $n << p$, at most the first $n$ eigenvalues are non zero. Hence, there is no need to calculate the other $p-n$ zero eigenvalues. When center is TRUE, the eigenvalues of the covariance matrix are calculated. When both the center and scale is TRUE the eigenvalues of the correlation matrix are calculated. One or more eigenvectors (towards the end) will be 0. In general the signs might be the opposite than R’s, but this makes no difference. We use the `Crossprod` instead of the relevant built-in function. The higher the dimensions of the matrix are the faster this function becomes.
Empirical and exponential empirical likelihood tests for one sample

Value
- A list including:
  - values: A vector with the n (or first k) eigenvalues. The divisor in the crossproduc matrix is n-1 and not n.
  - vectors: A matrix of $p \times n$ or $p \times k$ eigenvectors.

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

See Also
- rmdp

Examples
```r
x <- matrnorm(40, 100)
a <- hd.eigen(x, FALSE, FALSE)
b <- prcomp(x, center = FALSE, scale = FALSE)
a
b$sdev^2
x <- NULL
```

Description
Empirical and exponential empirical likelihood tests for one sample.

Usage
```r
eel.test1(x, mu, tol = 1e-09, logged = FALSE)
el.test1(x, mu, tol = 1e-07, logged = FALSE)
```

Arguments
- x: A numerical vector.
- mu: The hypothesised mean value.
- tol: The tolerance value to stop the iterations of the Newton-Raphson.
- logged: Should the logarithm of the p-value be returned? TRUE or FALSE.
Empirical and exponential empirical likelihood tests for two samples

Details

Exponential empirical likelihood is a non-parametric method. In this case we use it as the non-parametric alternative to the t-test. Newton-Raphson is used to maximise the log-likelihood ratio test statistic. In the case of no solution, NULL is returned. Despite the function having been written in R, it is pretty fast. As for the empirical likelihood ratio test, there is a condition for the range of possible values of \( \mu \). If \( \mu \) is outside this range it is rejected immediately.

Value

| iters | The number of iterations required by the Newton-Raphson algorithm. If no convergence occurred this is NULL. This is not returned for the empirical likelihood ratio test. |
| info | A vector with three elements, the value of the \( \lambda \), the likelihood ratio test statistic and the relevant p-value. If no convergence occurred, the value of the \( \lambda \) before is becomes NA, the value of test statistic is \( 10^5 \) and the p-value is 0. No convergence can be interpreted as rejection of the hypothesis test. |
| \( p \) | The estimated probabilities, one for each observation. If no convergence occurred this is NULL. |

Author(s)

Michail Tsagris

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

References


See Also

ttest, ttest1

Examples

```r
x <- rnorm(500)
system.time(a1 <- eel.test1(x, 0) )
system.time(a2 <- el.test1(x, 0) )
```

---

Empirical and exponential empirical likelihood tests for two samples

Empirical and exponential empirical likelihood tests for two samples

Description

Empirical and exponential empirical likelihood tests for two samples.
Empirical and exponential empirical likelihood tests for two samples

Usage

eel.test2(x, y, tol = 1e-09, logged = FALSE)
el.test2(x, y, tol = 1e-07, logged = FALSE)

Arguments

x A numerical vector.
y Another numerical vector.
tol The tolerance value to stop the iterations of the Newton-Raphson.
logged Should the logarithm of the p-value be returned? TRUE or FALSE.

Details

Empirical and exponential empirical likelihood are two non parametric hypothesis testing methods. We can use them as non parametric alternatives to the t-test. Newton-Raphson is used to maximise the log-likelihood ratio test statistic. In the case of no solution, NULL is returned.

Value

iters The number of iterations required by the Newton-Raphson algorithm. If no convergence occured this is NULL.
info A vector with three elements, the value of the \( \lambda \), the likelihood ratio test statistic and the relevant p-value. If no convergence occured, the value of the \( \lambda \) before is becomes NA, the value of test statistic is \( 10^5 \) and the p-value is 0. No convergence can be interpreted as rejection of the hypothesis test.
p1 The estimated probabilities, one for each observation for the first sample. If no convergence occured this is NULL.
p2 The estimated probabilities, one for each observation for the second sample. If no convergence occured this is NULL.

Author(s)

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

References


See Also

ftests, ttests, ttest
Energy distance between matrices

Examples

```r
x <- rnorm(200)
y <- rnorm(300)
system.time( eel.test2(x, y) )
system.time( el.test2(x, y) )
```

Description

Energy distance between matrices.

Usage

```r
edist(x, y=NULL)
```

Arguments

- `x` A matrix with numbers or a list with matrices.
- `y` A second matrix with data. The number of columns of `x` and `y` must match. The number of rows can be different.

Details

This calculates the energy distance between two matrices. It will work even for tens of thousands of rows, it will just take some time. See the references for more information. If you have many matrices and want to calculate the distance matrix, then put them in a list and use eDist.

Value

If "x" is matrix, a numerical value, the energy distance. If "x" is list, a matrix with all pairwise distances of the matrices.

Author(s)

Manos Papadakis

R implementation and documentation: Manos Papadakis <papadakm95@gmail.com>.

References

See Also
dvar, total.dist, total.dista, Dist, dista

Examples

```r
x <- as.matrix( iris[1:50, 1:4] )
y <- as.matrix( iris[51:100, 1:4] )
res<-edist(x, y)
z <- as.matrix(iris[101:150, 1:4])
a <- list()
a[[ 1 ]] <- x
a[[ 2 ]] <- y
a[[ 3 ]] <- z
res<-edist(a)
```

```
x<-y<-z<-a<-NULL
```

Equality of objects

Description
Equality of objects.

Usage

```
all_equals(x,y,round_digits = FALSE,without_attr=FALSE,fast_result=FALSE)
```

Arguments

- `x`: A Matrix, List, Dataframe or Vector.
- `y`: A Matrix, List, Dataframe or Vector.
- `round_digits`: The digit for rounding numbers.
- `without_attr`: A boolean value (TRUE/FALSE) for deleting attributes. Be careful although because some attributes are very important for you item.
- `fast_result`: A boolean value (TRUE/FALSE) for using just identical. But you can combine only with round_digits argument.

Value
A boolean (TRUE/FALSE) value which represents if the items x and y are equal.

Author(s)
Manos Papadakis
R implementation and documentation: Manos Papadakis <papadakm95@gmail.com>.
Estimation of an AR(1) model

See Also

Match, mvbetas, correls, univglms, colsums, colVars

Examples

```r
x <- matrix(rnorm(100 * 100), ncol = 100)
y <- matrix(rnorm(100 * 100), ncol = 100)
all_equals(x, y)
all_equals(x, x)
```

Description

Estimation of an AR(1) model.

Usage

```r
ar1(y, method = "cmle")
colar1(y, method = "cmle")
```

Arguments

- `y` For the case of `ar1` this is a vector of time series. For the case of `colar1` this is a matrix where each column represents a time series.
- `method` This can be either "cmle" for conditional maximum likelihood or "yw" for the Yule-Walker equations.

Details

Instead of the classical MLE for the AR(1) model which requires numerical optimisation (Newton-Raphson for example) we estimate the parameters of the AR(1) model using conditional maximum likelihood. This procedure is described in Chapter 17 in Lee (2006). In some, it assumes that the first observation is deterministic and hence conditioning on that observation, there is a closed form solution for the parameters. The second alternative is to use the method of moments and hence the Yule-Walker equations.

Value

- `param` For the case of `ar1` this is a vector with three elements, the constant term, the \( \phi \) term (lag coefficient) and the variance. For the case of `colar1` this is a matrix with three columns, each of which carries the same aforementioned elements.
Estimation of the Box-Cox transformation

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

References

See Also
rm.lines, varcomps.mle, rm.anovas

Examples
y <- as.vector(lh)
ar1(y)
ar(y, FALSE, 1, "ols")

ar1(y, method = "yw")
ar(y, FALSE, 1, "yw")

a1 <- colar1(cbind(y, y))
b1 <- colar1(cbind(y, y), method = "yw")

Estimation of the Box-Cox transformation

Description
Estimation of the Box-Cox transformation.

Usage
bc(x, low = -1, up = 1)

Arguments
x A numerical vector with strictly positive values.
low The lowest value to search for the best \( \lambda \) parameter.
up The highest value to search for the best \( \lambda \) parameter.

Details
The functions estimates the best \( \lambda \) in the Box-Cox power transformation.
Exact t-test for 2 independent samples

Value

The optimal value of \( \lambda \).

Author(s)

Michail Tsagris

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

References


See Also

correls, auc

Examples

```r
x <- exp(rnorm(1000))
res <- bc(x)
```

Description

Exact t-test for 2 independent samples.

Usage

`exact.ttest2(x, y)`

Arguments

- `x` A numerical vector with the data.
- `y` A numerical vector with the data.

Details

This function performs an exact t-test. With few observations, permutation or bootstrap calculation of the p-value is advisable. However, with even fewer observations, one can perform all possible permutations and calculate the exact p-value. This is what this function does. BUT, pay attention, as this works with few samples. If for example each sample contains 15 numbers, you will need a lot of memory (more than 17 GB) for this function to work. the reason is that we create the matrix with all possible permutations first and then perform the two-sample t-test.
Exponential empirical likelihood for a one sample mean vector hypothesis testing

Value

A vector with the number of permutations, test statistic and the permutation based p-value.

Author(s)

Michail Tsagris and Manos Papadakis

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

References


See Also

boot.ttest2, ttest2, ftest

Examples

```r
x <- rnorm(7)
y <- rnorm(7)
res <- exact.ttest2(x, y)
```

Description

Exponential empirical likelihood for a one sample mean vector hypothesis testing.

Usage

```r
mv.eeltest1(x, mu, tol = 1e-06)
```

Arguments

- `x`: A matrix containing Euclidean data.
- `mu`: The hypothesized mean vector.
- `tol`: The tolerance value used to stop the Newton-Raphson algorithm.

Details

Multivariate hypothesis test for a one sample mean vector. This is a non parametric test and it works for univariate and multivariate data. The p-value is currently computed only asymptotically (no bootstrap calibration at the moment).
Exponential empirical likelihood hypothesis testing for two mean vectors

Value

A list including:

- `p`: The estimated probabilities.
- `lambda`: The value of the Lagrangian parameter $\lambda$.
- `iters`: The number of iterations required by the newton-Raphson algorithm.
- `info`: The value of the log-likelihood ratio test statistic along with its corresponding p-value.

Author(s)

Michail Tsagris

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

References


See Also

`james, mv.eeltest2`

Examples

```R
x <- Rfast::rmvnorm(100, numeric(10), diag( rexp(10, 0.5) ) )
res<-.mv.eeltest1(x, numeric(10) )
```

Description

Exponential empirical likelihood hypothesis testing for two mean vectors.

Usage

`mv.eeltest2(y1, y2, tol = 1e-07, R = 0)`
Exponential empirical likelihood hypothesis testing for two mean vectors

Arguments

\begin{itemize}
  \item \texttt{y1} \hspace{1cm} A matrix containing the Euclidean data of the first group.
  \item \texttt{y2} \hspace{1cm} A matrix containing the Euclidean data of the second group.
  \item \texttt{tol} \hspace{1cm} The tolerance level used to terminate the Newton-Raphson algorithm.
  \item \texttt{R} \hspace{1cm} If R is 0, the classical chi-square distribution is used, if R = 1, the corrected chi-square distribution (James, 1954) is used and if R = 2, the modified F distribution (Krishnamoorthy and Yanping, 2006) is used.
\end{itemize}

Details

Exponential empirical likelihood is a non-parametric hypothesis testing procedure for one sample. The generalisation to two (or more samples) is via searching for the mean vector that minimises the sum of the two test statistics.

Value

A list including:

\begin{itemize}
  \item \texttt{test} \hspace{1cm} The empirical likelihood test statistic value.
  \item \texttt{modif.test} \hspace{1cm} The modified test statistic, either via the chi-square or the F distribution.
  \item \texttt{pvalue} \hspace{1cm} The p-value.
  \item \texttt{iters} \hspace{1cm} The number of iterations required by the newton-Raphson algorithm.
  \item \texttt{mu} \hspace{1cm} The estimated common mean vector.
\end{itemize}

Author(s)

Michail Tsagris

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

References


Fast and general - untyped representation of a factor variable

See Also
james,mv.eeltest1

Examples
res<-mv.eeltest2( as.matrix(iris[1:25, 1:4]), as.matrix(iris[26:50, 1:4]), R = 0 )
res<-mv.eeltest2( as.matrix(iris[1:25, 1:4]), as.matrix(iris[26:50, 1:4]), R = 1 )

Description
Fast and general representation of a factor variable.

Usage
ufactor(x)
## S3 method for class 'ufactor'
x[i]
## S3 method for class 'ufactor'
print(x,...)

Arguments
x
A vector with data.
i
An integer value/vector which is the index/indices to the element you want to access.
... Anything the user want.

Details
This is a general implementation of factor structure. For access the fields of a "ufactor" use the "$" operator.

Value
An object of class "ufactor". This object holds 2 fields:
levels: the levels of the variable in his initial type values: the values of the variable in his initial type

Author(s)
Manos Papadakis
R implementation and documentation: and Manos Papadakis <papadakm95@gmail.com>.
FBED variable selection method using the correlation

See Also
colVars, factor

Examples

```r
x <- rnorm(10)
R.factor<- as.factor(x)
Rfast.factor <- ufactor(x)

identical(levels(R.factor),Rfast.factor$levels) # TRUE
identical(as.numeric(R.factor),Rfast.factor$values) # TRUE
x<-R.factor<-Rfast.factor<-NULL
```

FBED variable selection method using the correlation

FBED variable selection method using the correlation

Description

FBED variable selection method using the correlation.

Usage

```r
cor.fbed(y, x, ystand = TRUE, xstand = TRUE, alpha = 0.05, K = 0)
```

Arguments

- **y**: The response variable, a numeric vector.
- **x**: A matrix with the data, where the rows denote the samples and the columns are the variables.
- **ystand**: If this is TRUE the response variable is centered. The mean is subtracted from every value.
- **xstand**: If this is TRUE the independent variables are standardised.
- **alpha**: The significance level, set to 0.05 by default.
- **K**: The number of times to repeat the process. The default value is 0.

Details

FBED stands for Forward Backward with Early Dropping. It is a variation of the classical forward selection, where at each step, only the statistically significant variables carry on. The rest are dropped. The process stops when no other variables can be selected. If K = 1, the process is repeated testing sequentially again all those that have not been selected. If K > 1, then this is repeated.

In the end, the backward selection is performed to remove any falsely included variables. This backward phase has not been implemented yet.
Find element

Value

A list including:

- `runtime`: The duration of the process.
- `res`: A matrix with the index of the selected variable, their test statistic value and the associated p-value.
- `info`: A matrix with two columns. The cumulative number of variables selected and the number of tests for each value of K.

Author(s)

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

References


See Also

cor.fsreg, ompr, correls, fs.reg

Examples

```r
x <- matrnorm(100, 100)
y <- rnorm(100)
a <- cor.fbed(y, x)
a
x <- NULL
```

Find element

Description

Search a value in an unordered vector.

Usage

```r
is_element(x, key)
```

Arguments

- `x`: A vector or matrix with the data.
- `key`: A value to check if exists in the vector `x`. 
Find the given value in a hash table

Details
Find if the key exists in the vector and return returns TRUE/FALSE if the value is been found. If the vector is unordered it is fast but if the vector is ordered then use binary_search. The functions is written in C++ in order to be as fast as possible.

Value
TRUE/FALSE if the value is been found.

Author(s)
Manos Papadakis
R implementation and documentation: Manos Papadakis <papadakm95@gmail.com>.

See Also
binary_search (built-in R function)

Examples
```r
x <- rnorm(500)
key <- x[50]
b <- is_element(x, key)
```

Description
Find the given value in a hash table or list.

Usage
```r
hash.find(x, key)
```

Arguments
- **x**: A hash table or list.
- **key**: The key for searching the table.

Details
This function search the given key.

Value
If the given key exists return its value else returns 0.
Fitted probabilities of the Terry-Bradley model

Author(s)

Manos Papadakis

R implementation and documentation: Manos Papadakis <papadakm95@gmail.com>

See Also

hash.list

Examples

x <- hash.list(letters,c(1:26))
value <- hash.find(x,"a")
x["a"]==value

Fitted probabilities of the Terry-Bradley model

Fitted probabilities of the Terry-Bradley model

Description

Fitted probabilities of the Terry-Bradley model.

Usage

btmprobs(x, tol = 1e-09)

Arguments

x
A numerical square, usually not symmetric, matrix with discrete valued data. Each entry is a frequency, to give an example, the number of wins. \(x[i, j]\) is the number of wins of home team \(i\) against guest team \(j\). \(x[j, i]\) is the number of wins of home team \(i\) against guest team \(j\).

tol
The tolerance level to terminate the iterative algorithm.

Details

It fits a Bradley-Terry model to the given matrix and returns the fitted probabilities only.

Value

A list including:

iters
The number of iterations required.

probs
A vector with probabilities which sum to 1. This is the probability of win for each item (or team in our hypothetical example).
Fitting a Dirichlet distribution via Newton-Rapshon

Author(s)

Michail Tsagris

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

References


See Also

g2tests,poisson.anova,anova,poisson_only,poisson.mle

Examples

```r
x <- matrix( rpois(10 * 10, 10), ncol = 10) ## not the best example though
res<-btmprobs(x)
```

Description

Fitting a Dirichlet distribution via Newton-Rapshon.

Usage

```r
diri.nr2(x, type = 1, tol = 1e-07)
```

Arguments

- `x`: A matrix containing the compositional data. Zeros are not allowed.
- `type`: Type 1 uses a vectorised version of the Newton-Raphson (Minka, 2012). In high dimensions this is to be preferred. If the data are too concentrated, regardless of the dimensions, this is also to be preferred. Type 2 uses the regular Newton-Raphson, with matrix multiplications. In small dimensions this can be considerably faster.
- `tol`: The tolerance level indicating no further increase in the log-likelihood.
Floyd-Warshall algorithm

Details

Maximum likelihood estimation of the parameters of a Dirichlet distribution is performed via Newton-Raphson. Initial values suggested by Minka (2012) are used.

Value

A list including:

- loglik: The value of the log-likelihood.
- param: The estimated parameters.

Author(s)

Michail Tsagris and Manos Papadakis

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

References


See Also

beta.mle

Examples

```r
x <- matrix(rgamma(100 * 4, c(5, 6, 7, 8), 1), ncol = 4)
x <- x / rowsums(x)
res <- diri.nr2(x)
```

---

**Floyd-Warshall algorithm**

*Floyd-Warshall algorithm for shortest paths in a directed graph*

---

Description

Floyd-Warshall algorithm for shortest paths in a directed graph.

Usage

floyd(x)
Floyd-Warshall algorithm

Arguments

\(x\)  
The adjacency matrix of a directed graph. A positive number (including) in \(x[i, j]\) indicates that there is an arrow from \(i\) to \(j\) and it also shows the cost of going from \(i\) to \(j\). Hence, the algorithm will find not only the shortest path but also the one with the smallest cost. A value of NA means that there is no path. Put positive number only, as negative will cause problems.

Details

The Floyd-Warshall algorithm is designed to find the shortest path (if it exists) between two nodes in a graph.

Value

A matrix, say \(z\), with 0 and positive numbers. The elements denote the length of the shortest path between each pair of points. If \(z[i, j]\) is zero it means that there is no cost from \(i\) to \(j\). If \(z[i, j]\) has a positive value it means that the length of going from \(i\) to \(j\) is equal to that value.

Author(s)

John Burkardt (C++ code)

Ported into R and documentation: Manos Papadakis <papadakm95@gmail.com>.

References


https://en.wikipedia.org/wiki/Floyd

See Also

\(\text{colSort, rowSort}\)

Examples

\[
x <- \text{matrix(NA, 10, 10)}
x[\text{sample(1:100, 10)] <- \text{rpois(10, 3)}
\]
\[
\text{res} <- \text{floyd(x)}
\]
Variable selection in generalised linear regression models with forward selection

Usage

fs.reg(y, ds, sig = 0.05, tol = 2, type = "logistic")

Arguments

y
  The dependent variable. This can either be a binary numeric (0, 1) or a vector with integers (numeric or integer class), count data. The first case is for the binary logistic regression and the second for the Poisson regression.

ds
  The dataset; provide a matrix where columns denote the variables and the rows the observations. The variables must be continuous, no categorical variables are accepted.

sig
  Significance level for assessing the p-values significance. Default value is 0.05.

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.

type
  If you have a binary dependent variable, put "logistic" or "quasibinomial". If you have percentages, values between 0 and 1, including 0 and or 1, use "quasibinomial" as well. If you have count data put "poisson".

Details

The classical forward regression is implemented. The difference is that we have an extra step of check. Even if a variable is significant, the BIC of the model (with that variable) is calculated. If the decrease from the previous BIC (of the model without this variable) is less than a prespecified by the user value (default is 2) the variable will enter. This way, we guard somehow against over-fitting.

Value

A matrix with for columns, the selected variables, the logarithm of their p-value, their test statistic and the BIC of the model with these variables included. If no variable is selected, the matrix is empty.

Author(s)

Marios Dimitriadis

Documentation: Marios Dimitriadis <kmdimitriadis@gmail.com>.
G-square and Chi-square test of conditional independence

See Also
cor.fsreg, logistic_only, poisson_only, glm_logistic, glm_poisson

Examples

## Not run:
set.seed(123)

# simulate a dataset with continuous data
x <- matrnorm(100, 50)
y <- rpois(100, 10)
a <- fs.reg(y, x, sig = 0.05, tol = 2, type = "poisson")
x <- NULL
## End(Not run)

---

G-square and Chi-square test of conditional independence

G-square test of conditional independence

Description

G-square test of conditional independence with and without permutations.

Usage

g2Test(data, x, y, cs, dc)
g2Test_perm(data, x, y, cs, dc, nperm)
chi2Test(data, x, y, cs, dc)

Arguments

data  A numerical matrix with the data. The minimum must be 0, otherwise the function can crash or will produce wrong results. The data must be consecutive numbers.
x  A number between 1 and the number of columns of data. This indicates which variable to take.
y  A number between 1 and the number of columns of data (other than x). This indicates the other variable whose independence with x is to be tested.
cs  A vector with the indices of the variables to condition upon. It must be non zero and between 1 and the number of variables. If you want unconditional independence test see g2Test_univariate and g2Test_univariate_perm. If there is an overlap between x, y and cs you will get 0 as the value of the test statistic.
dc  A numerical value equal to the number of variables (or columns of the data matrix) indicating the number of distinct, unique values (or levels) of each variable. Make sure you give the correct numbers here, otherwise the degrees of freedom will be wrong.

nperm  The number of permutations. The permutations test is slower than without permutations and should be used with small sample sizes or when the contingency tables have zeros. When there are few variables, R’s "chisq.test" function is faster, but as the number of variables increase the time difference with R’s procedure becomes larger and larger.

Details

The functions calculates the test statistic of the $G^2$ test of conditional independence between x and y conditional on a set of variable(s) cs.

Value

A list including:

- statistic  The $G^2$ or chi$^2$ test statistic.
- df  The degrees of freedom of the test statistic.
- x  The row or variable of the data.
- y  The column or variable of the data.

Author(s)

Giorgos Borboudakis. The permutation version used a C++ code by John Burkardt.

R implementation and documentation: Manos Papadakis <papadakm95@gmail.com>.

References


See Also

g2Test_univariate, g2Test_univariate_perm, correls, univglm

Examples

nvalues <- 3
nvars <- 10
nsamples <- 5000
data <- matrix( sample( 0:(nvalues - 1), nvars * nsamples, replace = TRUE ), nsamples, nvars )
dc <- rep(nvalues, nvars)
res<-g2Test( data, 1, 2, 3, c(3, 3, 3) )
res<-g2Test_perm( data, 1, 2, 3, c(3, 3, 3), 1000 )
Gamma regression with a log-link

**Description**
Gamma regression with a log-link.

**Usage**
```
    gammareg(y, x, tol = 1e-07, maxiters = 100)
    gammacon(y, tol = 1e-08, maxiters = 50)
```

**Arguments**
- **y**: The dependent variable, a numerical variable with non-negative numbers.
- **x**: A matrix or data.frame with the independent variables.
- **tol**: The tolerance value to terminate the Newton-Raphson algorithm.
- **maxiters**: The maximum number of iterations that can take place in the regression.

**Details**
The `gammareg` fits a Gamma regression with a log-link. The `gammacon` fits a Gamma regression with a log link with the intercept only (`glm(y ~ 1, Gamma(log))`).

**Value**
A list including:
- **deviance**: The deviance value.
- **phi**: The dispersion parameter ($\phi$) of the regression. This is necessary if you want to perform an F hypothesis test for the significance of one or more independent variables.
- **be**: The regression coefficient(s).
- **info**: The number of iterations, the deviance and the dispersion parameter.

**Author(s)**
Michaël Tsagris
R implementation and documentation: Michaël Tsagris <mtsagris@uoc.gr>.
Gaussian regression with a log-link

References

See Also
`gammaregs, normlog.reg, invgauss.reg`

Examples

```r
y <- abs(rnorm(100))
x <- matrix(rnorm(100 * 2), ncol = 2)
mod <- glm(y ~ x, family = Gamma(log))
res <- summary(mod)
## Not run:
res <- gammareg(y, x)
## End(Not run)
mod <- glm(y ~ 1, family = Gamma(log))
res <- summary(mod)
res <- gammacon(y)
```

Description

Gaussian regression with a log-link.

Usage

```r
normlog.reg(y, x, tol = 1e-07, maxiters = 100)
```

Arguments

- `y`: The dependent variable, a numerical variable with non-negative numbers.
- `x`: A matrix or data.frame with the independent variables.
- `tol`: The tolerance value to terminate the Newton-Raphson algorithm.
- `maxiters`: The maximum number of iterations that can take place in the regression.

Details

A Gaussian regression with a log-link is fitted.
Generates random values from a normal and puts them in a matrix

Value
A list including:

- \( i \)  The number of iterations required by the Newton-Raphson
- \( \text{loglik} \)  The log-likelihood value.
- \( \text{deviance} \)  The deviance value.
- \( \text{be} \)  The regression coefficients

Author(s)
Stefanos Fafalios
R implementation and documentation: Stefanos Fafalios <stefanosfafalios@gmail.com>

See Also

- \texttt{normlog.regs}, \texttt{score.glms}, \texttt{prop.regs}, \texttt{allbetas}

Examples

```r
## Not run:
y <- abs( rnorm(100) )
x <- matrix( rnorm(100 * 2), ncol = 2)
a <- normlog.reg(y, x)
b <- glm(y ~ x, family = gaussian(log) )
summary(b)
a
## End(Not run)
```

Description
Generates random values from a normal and puts them in a matrix.

Usage

```r
matrnorm(n, p, seed = NULL)
```

Arguments

- \( n \)  The sample size, the number of rows the matrix will have.
- \( p \)  The dimensionality of the data, the number of columns of the matrix.
- \( \text{seed} \)  If you want the same to be generated again use a seed for the generator, an integer number.
Get specific columns/rows of a matrix

Details

How many times did you have to simulated data from a (standard) normal distribution in order to test something? For example, in order to see the speed of `logistic_only`, one needs to generate a matrix with predictor variables. The same is true for other similar functions. In `sftests`, one would like to examine the type I error of this test under the null hypothesis.

By using the Ziggurat method of generating standard normal variates, this function is really fast when you want to generate big matrices.

Value

An n x p matrix with data simulated from a standard normal distribution.

Author(s)

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

See Also

rvmf, Rnorm, rmvnorm, rvonmises

Examples

```r
x <- matrnorm(100, 100)
```

---

Get specific columns/rows of a matrix

Description

Get specific columns/rows of a matrix.

Usage

```r
columns(x, indices)
rows(x, indices)
```

Arguments

- `x` A matrix with data.
- `indices` An integer vector with the indices.

Value

A matrix with the specific columns/rows of argument indices.
Hash - Pair function

Author(s)

Manos Papadakis

R implementation and documentation: Manos Papadakis <papadakm95@gmail.com>.

See Also

rowMins, rowFalse, nth, colrange, colMedians, colVars, colSort, rowSort, rowTrue

Examples

x <- matrix(runif(100*100),100,100)
indices = sample(1:100,50)
all.equal(x[,indices],columns(x,indices))
all.equal(x[indices,],rows(x,indices))

x<-indices<-NULL

Description

Hash - Pair function.

Usage

hash.list(key,x)

Arguments

key
The keys of the given values.
x
The values.

Details

This function pairs each item of of key and value make a unique hash table.

Value

Returns the hash-list table.

Author(s)

Manos Papadakis

R implementation and documentation: Manos Papadakis <papadakm95@gmail.com>
Hash object

See Also

hash.find

Examples

```r
x <- hash.list(letters, c(1:26))
x[["a"]]<-1
```

Hash object

Hash object

Description

Hash object.

Usage

```r
Hash(keys=NULL, values=NULL)
Hash.key.multi(x, ..., sep = " ")
## S3 replacement method for class 'Hash'
x[... , sep = " "] <- value
## S3 method for class 'Hash'
x[... , sep = " "]
## S3 method for class 'Hash'
print(x, ...)
## S3 method for class 'Hash'
length(x)
```

Arguments

- `x`: A Hash object, using Hash function.
- `values`: A vector with the values you want to store.
- `value`: The values you want to store.
- `keys`: A vector with keys for each values.
- `sep`: A character value using to separate the multiple keys for each value.
- `...`: One or more values for access or find elements.

Details

If you want to delete a key just insert the global variable "Rfast:::delete".

Hash: Create Hash object where every key has a value. Specify the type from the begginning (for speed). Use the argument "type" with one of the values "new.env, logical, character, integer, numeric". Hash.key.multi: search if key exists. If the keys are multiple, then use the argument "substr" to search inside each multiple for the specific key.
Hash object to a list object

Value

A Hash object.

Author(s)

Manos Papadakis

R implementation and documentation: Manos Papadakis <papadakm95@gmail.com>.

See Also

hash.list, hash.find

Examples

```r
x <- Hash(rnorm(10), sample(1:10))

x[1,2,13] <- 0.1234 # insert value using multi key. the same as x["1 2 13"] <- 0.1234
x[1,2,3] <- 15 # insert value using multi key. the same as x["1 2 3"] <- 15

Hash.key.multi(x,"1")  
x # print Hash object using S3 generic

#x[1,2,3] <- Rfast:::delete # delete multi key. the same as x["1 2 3"] <- NULL
length(x)
```

Description

Hash object to a list object.

Usage

```r
hash2list(x, sorting = FALSE)
```

Arguments

- **x**: A hash table with two parts, the keys (number(s) as string) and the key values (a single number).
- **sorting**: This is if you want the numbers in the keys sorted. The default value is FALSE.

Details

For every key, there is a key value. This function creates a list and puts every pair of keys and value in a component of a list.
High dimensional MCD based detection of outliers

Value

A list whose length is equal to the size of the hash table.

Author(s)

Manos Papadakis

R implementation and documentation: Manos Papadakis <papadakm95@gmail.com>.

See Also

hash.list, hash.find

Examples

x=list("1 2 4 3"=2.56,"2.34 1.05"=2)
res<-hash2list(x)
res<-hash2list(x,TRUE)

Description

High dimensional MCD based detection of outliers.

Usage

rmdp(y, alpha = 0.05, itertime = 100)

Arguments

y A matrix with numerical data with more columns (p) than rows (n), i.e. n<p.
alpha The significance level, i.e. used to decide whether an observation is said to be considered a possible outlier. The default value is 0.05.
iterime The number of iterations the algorithm will be ran. The higher the sample size, the larger this number must be. With 50 observations in R^200 maybe this has to be 1000 in order to produce stable results.

Details

High dimensional outliers (n<p) are detected using a properly constructed MCD. The variances of the variables are used and the determinant is simply their product.
Hypothesis test for the distance correlation

Value

A list including: runtime = runtime, dis = dis, wei = wei

- runtime: The duration of the process.
- dis: The final estimated Mahalanobis type normalised distances.
- wei: A boolean variable vector specifying whether an observation is "clean" (TRUE) or a possible outlier (FALSE).
- cova: The estimated covariance matrix.

Author(s)

Initial R code: Changliang Zou <nk.chlzou@gmail.com> R code modifications: Michail Tsagris <mtsagris@uoc.gr> C++ implementation: Manos Papadakis <papadakm95@gmail.com> Documentation: Michail Tsagris <mtsagris@uoc.gr> and Changliang Zhou <nk.chlzou@gmail.com>

References


See Also

colmeans, colVars, colMedians

Examples

```r
x <- matrix(rnorm(50 * 400), ncol = 400)
a <- rmdp(x, itertime = 500)
x <- a <- NULL
```

Description

Hypothesis test for the distance correlation.

Usage

dcor.ttest(x, y, logged = FALSE)
Hypothesis test for the distance correlation

Arguments

x A numerical matrix.

y A numerical matrix.

logged Do you want the logarithm of the p-value to be returned? If yes, set this to TRUE.

Details

The bias corrected distance correlation is used. The hypothesis test is whether the two matrices are independent or not. Note, that this test is size correct as both the sample size and the dimensionality goes to infinity. It will not have the correct type I error for univariate data or for matrices with just a couple of variables.

Value

A vector with 4 elements, the bias corrected distance correlation, the degrees of freedom, the test statistic and its associated p-value.

Author(s)

Manos Papadakis

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

References


See Also

bcdcor, dcov, edist

Examples

```r
x <- as.matrix(iris[1:50, 1:4])
y <- as.matrix(iris[51:100, 1:4])
res <- dcor.ttest(x, y)
```
Hypothesis test for two means of percentages

**Description**

Hypothesis test for two means of percentages.

**Usage**

```r
percent.ttest(x, y, logged = FALSE)
```

**Arguments**

- `x`: A numerical vector with the percentages of the first sample. Any value between 0 and 1 (inclusive) is allowed.
- `y`: A numerical vector with the percentages of the first sample. Any value between 0 and 1 (inclusive) is allowed.
- `logged`: Should the p-values be returned (FALSE) or their logarithm (TRUE)?

**Details**

This is the `prop.reg` but with a single categorical predictor which has two levels only. It is like a t-test for the means of two samples having percentages.

**Value**

A vector with three elements, the phi parameter, the test statistic and its associated p-value.

**Author(s)**

Michail Tsagris

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

**References**


**See Also**

`link{percent.ttests}, prop.reg, ttest2, ftest`
Hypothesis test for von Mises-Fisher distribution over Kent distribution

Examples

```r
x <- rbeta(100, 3, 1)
y <- rbeta(100, 7.5, 2.5)
res <- percent.ttest(x, y)
```

Description

The null hypothesis is whether a von Mises-Fisher distribution fits the data well, and the alternative is that the Kent distribution is more suitable.

Usage

```r
fish.kent(x, logged = FALSE)
```

Arguments

- `x`: A numeric matrix containing the data as unit vectors in Euclidean coordinates.
- `logged`: If you want the logarithm of the p-value to be returned set this to TRUE.

Details

Essentially it is a test of rotational symmetry, whether Kent’s ovalness parameter (beta) is equal to zero. This works for spherical data only.

Value

A vector with two elements, the value of the test statistic and its associated p-value.

Author(s)

Michail Tsagris

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

References


See Also

vmf.mle, iag.mle
Hypothesis testing between two skewness or kurtosis coefficients

Examples

```r
x <- rvmf(100, rnorm(3), 15)
res <- fish.kent(x)
x <- NULL
```

Description

Hypothesis testing between two skewness or kurtosis coefficients.

Usage

```r
skew.test2(x, y)
kurt.test2(x, y)
```

Arguments

- `x` A numerical vector with data.
- `y` A numerical vector with data, not necessarily of the same size.

Details

The skewness of kurtosis coefficients between two samples are being compared.

Value

A vector with the test statistic and its associated p-value.

Author(s)

Klio Lakiotaki

R implementation and documentation: Klio Lakiotaki <kliolak@gmail.com>.

References


See Also

- `skew`, `colskewness`, `colmeans`, `colVars`, `colMedians`
Index of the columns of a data.frame which are a specific type

Examples

```r
x <- rgamma(150, 1, 4)
y <- rgamma(100, 1, 4)
res <- skew.test2(x, y)
res <- kurt.test2(x, y)
```

Description

Index of the columns of a data.frame which are a specific type.

Usage

```r
which.is(x, method = "factor")
```

Arguments

- `x`: A data.frame where some columns are expected to be factor variables.
- `method`: A character value about the type. One of "numeric", "factor", "integer", "logical".

Details

The function is written in C++ and this is why it is very fast.

Value

A vector with the column indices which are factor variables. If there are no factor variables it will return an empty vector.

Author(s)

Manos Papadakis <papadakm95@gmail.com>

R implementation and documentation: Manos Papadakis <papadakm95@gmail.com>.

See Also

nth, Match

Examples

```r
res <- which.is(iris)
```
Description

Insert/remove function names in/from the NAMESPACE file.

Usage

AddToNamespace(path.namespace, path.rfolder)
RemoveFromNamespace(path.namespace, files.to.remove)

Arguments

path.namespace  An full path to the NAMESPACE file.
path.rfolder    An full path to the directory the new files to be added are stored.
files.to.remove An character with the names of the functions to be removed from file NAMESPACE.

Details

AddToNameSpace: Reads the files that are exported in NAMESPACE and the functions that are inside rfolder (where R files are) and insert every function that is not exported. For that you must add the attribute "#[export]" above every function you wish to export. Also you can use the attribute "#[export s3]" for exporting S3methods. Finally, if you don’t want the program to read a file just add at the top of the file the attribute "#[dont read]".

RemoveFromNamespace: Remove every function, from argument "files.to.remove", from NAMESPACE.

Value

AddToNameSpace: Return the files that didn’t have the attribute "#[export]" or empty character vector if all the files was inserted.
RemoveFromNamespace: Return the files that could not be removed.

Author(s)

R implementation and documentation: Manos Papadakis <papadakm95@gmail.com>.

See Also

colShuffle, colVars, colmeans, read.directory
Inverse Gaussian regression with a log-link

Examples

```r
## Not run:
for example: path.namespace="C:\some_file\NAMESPACE" where is NAMESPACE file
path.rfolder="C:\some_file\R" where is R files are
system.time( a<-AddToNamespace(path.namespace,path.rfolder) )
if(length(a)==0){
print("all the files are inserted")
}else{
print("The new files that inserted are: \n")
a
}

system.time( a<-RemoveFromNamespace(path.namepace,c("a","b")) )
if(length(a)==0){
print("all the files are inserted")
}else{
print("The files that could not be deleted are: \n")
a
}
```

## End(Not run)

Inverse Gaussian regression with a log-link

Description

Inverse Gaussian regression with a log-link.

Usage

```r
invgauss.reg(y, x, tol = 1e-07, maxiters = 100)
```

Arguments

- `y`: The dependent variable, a numerical variable with non negative numbers.
- `x`: A matrix or data.frame with the independent variables.
- `tol`: The tolerance value to terminate the Newton-Raphson algorithm.
- `maxiters`: The maximum number of iterations that can take place in the regression.

Details

An inverse Gaussian regression with a log-link is fitted.
Inverse of a symmetric positive definite matrix

Value

A list including:

- **i**: The number of iterations required by the Newton-Raphson
- **loglik**: The log-likelihood value.
- **deviance**: The deviance value.
- **phi**: The dispersion parameter ($\phi$) of the regression. This is necessary if you want to perform an F hypothesis test for the significance of one or more independent variables.
- **be**: The regression coefficients

Author(s)

Michail Tsagris

R implementation and documentation: Stefanos Fafalios <mtsagris@uoc.gr>

References


See Also

- `invgauss.regs`, `normlog.reg`, `score.glms`

Examples

```r
## Not run:
y <- abs( rnorm(100) )
x <- matrix( rnorm(100 * 2), ncol = 2)
a <- invgauss.reg(y, x)
a
```

```r
## End(Not run)
```
Usage

\texttt{spdinv(A)}

Arguments

\texttt{A} \quad \text{A square positive definite matrix.}

Details

After calculating the Cholesky decomposition of the matrix we use this upper triangular matrix to invert the original matrix.

Value

The inverse of the input matrix.

Author(s)

Michail Tsagris

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

References


See Also

\texttt{cholesky, cova}

Examples

\begin{verbatim}
  s <- cova(as.matrix(iris[, 1:4]))
  res <- spdinv(s)
  res <- solve(s)
\end{verbatim}

Description

A way to traverse a list, data.frame, matrix or vector.
Usage

iterator(x, method="ceil", type="vector", by=1)
## S3 method for class 'iterator'
print(x,...)
## S3 replacement method for class 'iterator'
Elem(x) <- value
Elem(x)
Elem(x) <- value
## S3 method for class 'iterator'
Elem(x)
## S3 method for class 'iterator'
Elem(x)
## S3 method for class 'iterator'
x == y
## S3 method for class 'iterator'
x != y

Arguments

x A variable with any type, or iterator object.
value An value depending the method of the iterator.
y An iterator.
method Method of the iterator class. One of "ceil","col","row".
type One of "vector","matrix","data.frame","list".
by An integer value to iterate through element.
... Anything the user want.

Details

iterator: is an object that helps a programmer to traverse the given object.
print.iterator: print an object of class iterator.
"Elem<-": access to element and change the value.
Elem: access to element.

Value

An object of class "iterator". This object holds 4 fields:
copy: deep copy of iterator. end: get iterator tha have access to points to the last element. equals: equality of iterators nextElem: move iterator to point to the next element using argument "by".
prevElem: move iterator to point to the previous element using argument "by".

Author(s)

R implementation and documentation: Manos Papadakis <papadakm95@gmail.com>.

See Also

colShuffle, colVars, colmeans, read.directory
Examples

```r
y <- rnorm(100)
x <- iterator(y, method="ceil", type="vector", by=1)

s <- 0
while(x != x$end()){
  s <- s + Elem(x)
x$nextElem()
}

all.equal(s, sum(y))
```

Description

James test for testing the equality of two population mean vectors without assuming equality of the covariance matrices.

Usage

```r
james(y1, y2, a = 0.05, R = 1)
```

Arguments

- `y1` A matrix containing the Euclidean data of the first group.
- `y2` A matrix containing the Euclidean data of the second group.
- `a` The significance level, set to 0.05 by default.
- `R` If R is 1 the classical James test is returned. If R is 2 the MNV modification is implemented.

Details

Multivariate analysis of variance without assuming equality of the covariance matrices. The p-value can be calculated either asymptotically or via bootstrap. The James test (1954) or a modification proposed by Krishnamoorthy and Yanping (2006) is implemented. The James test uses a corrected chi-square distribution, whereas the modified version uses an F distribution.

Value

A list including:

- `note` A message informing the user about the test used.
- `mesoi` The two mean vectors.
The test statistic, the p-value, the correction factor and the corrected critical value of the chi-square distribution if the James test has been used or, the test statistic, the p-value, the critical value and the degrees of freedom (numerator and denominator) of the F distribution if the modified James test has been used.

Author(s)

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

References


See Also

mv.eeltest2

Examples

james( as.matrix(iris[1:25, 1:4]), as.matrix(iris[26:50, 1:4]), R = 1 )
james( as.matrix(iris[1:25, 1:4]), as.matrix(iris[26:50, 1:4]), R = 2 )

Description

k nearest neighbours algorithm (k-NN).

Usage

knn(xnew, y, x, k, dist.type = "euclidean", type = "C", method = "average", freq.option = 0, mem.eff = FALSE)

Arguments

xnew The new data, new predictor variable values. A matrix with numerical data.
y A vector with the response variable, whose values for the new data we wish to predict. This can be numerical data, factor or discrete, 0, 1, ... The latter two cases are for classification.
x The dataset. A matrix with numerical data.
**k nearest neighbours algorithm (k-NN)**

- **k**: The number of nearest neighbours to use. The number can either be a single value or a vector with multiple values.
- **dist.type**: The type of distance to be used. Either "euclidean" or "manhattan".
- **type**: If your response variable "y" is numerical data, then this should be "R" (regression). If "y" is in general categorical, factor or discrete set this argument to "C" (classification).
- **method**: In case you have regression (type = "R") you want a way to summarise the prediction. If you want to take the average of the responses of the k closest observations, type "average". For the median, type "median" and for the harmonic mean, type "harmonic".
- **freq.option**: If classification (type = "C") and ties occur in the prediction, more than one class has the same number of k nearest neighbours, in which case there are two strategies available: Option 0 selects the first most frequent encountered. Option 1 randomly selects the most frequent value, in the case that there are duplicates.
- **mem.eff**: Boolean value indicating a conservative or not use of memory. Lower usage of memory/Having this option on will lead to a slight decrease in execution speed and should ideally be on when the amount of memory in demand might be a concern.

**Details**

The concept behind k-NN is simple. Suppose we have a matrix with predictor variables and a vector with the response variable (numerical or categorical). When a new vector with observations (predictor variables) is available, its corresponding response value, numerical or category is to be predicted. Instead of using a model, parametric or not, one can use this ad hoc algorithm.

The k smallest distances between the new predictor variables and the existing ones are calculated. In the case of regression, the average, median or harmonic mean of the corresponding response values of these closest predictor values are calculated. In the case of classification, i.e. categorical response value, a voting rule is applied. The most frequent group (response value) is where the new observation is to be allocated.

**Value**

A matrix whose number of columns is equal to the size of k. If in the input you provided there is just one value of k, then a matrix with one column is returned containing the predicted values. If more than one value was supplied, the matrix will contain the predicted values for every value of k.

**Author(s)**

Marios Dimitriadis

R implementation and documentation: Marios Dimitriadis <kmdimitriadis@gmail.com>

**References**

http://statlink.tripod.com/id3.html

See Also
knn.cv, dirknn, logistic_only, fs.reg, cor.fsreg

Examples

# Simulate a dataset with continuous data
x <- as.matrix(iris[, 1:4])
y <- as.numeric(iris[, 5])
id <- sample(1:150, 120)
mod <- knn(x[-id, ], y[id], x[id, ], k = c(4, 5, 6), type = "C", mem.eff = FALSE)
mod # Predicted values of y for 3 values of k.
res<-table(mod[, 1], y[-id] ) # Confusion matrix for k = 4
res<-table(mod[, 2], y[-id] ) # Confusion matrix for k = 5
res<-table(mod[, 3], y[-id] ) # Confusion matrix for k = 6

Description

It classifies new observations to some known groups via the k-NN algorithm.

Usage

dirknn(xnew, x, y, k, type = "C", parallel = FALSE)

Arguments

xnew  The new data whose membership is to be predicted, a numeric matrix with unit vectors. In case you have one vector only make it a row vector (i.e. matrix with one row).
x     The data, a numeric matrix with unit vectors.
k     The number of nearest neighbours. It can also be a vector with many values.
y     A numerical vector representing the class or label of each vector of x. 1, 2, 3, and so on. It can also be a numerical vector with data in order to perform regression.
type  If your response variable y is numerical data, then this should be "R" (regression) or "WR" for distance weighted based nearest neighbours. If y is in general categorical set this argument to "C" (classification) or to "WC" for distance weighted based nearest neighbours.
Do you want the calculations to take place in parallel? The default value is FALSE.

Details

The standard algorithm is to keep the k nearest observations and see the groups of these observations. The new observation is allocated to the most frequent seen group. The non standard algorithm is to calculate the classical mean or the harmonic mean of the k nearest observations for each group. The new observation is allocated to the group with the smallest mean distance.

If you want regression, the predicted value is calculated as the average of the responses of the k nearest observations.

Value

A matrix with the predicted group(s). It has as many columns as the values of k.

Author(s)

Stefanos Fafalios

R implementation and documentation: Stefanos Fafalios <stefanosfafalios@gmail.com>

See Also

dirknn.cv,knn,vmf.mle,spml.mle

Examples

```r
x <- as.matrix(iris[, 1:4])
x <- x/sqrt(rowSums(x^2))
y <- as.numeric(iris[, 5])
a <- dirknn(x, x, y, k = 2:10)
```

Description

Limited number of eigenvalues and eigenvectors of a symmetric matrix.

Usage

eigen.sym(A, k, vectors = TRUE)

Arguments

A A symmetric matrix.

k The number of eigenvalues and eigenvectors to extract.

vectors A flag that indicates if the eigenvectors will be returned (default: vectors = True)
Linear models for large scale data

Details

The function calls the same function from the Armadillo library in C++. It is quite faster than R’s built in function “eigen” if the number of eigenvalues and eigenvectors (argument k) is small.

The k largest, in magnitude, eigenvalues are returned. Hence, if the matrix is not positive definite you may get negative eigenvalues as well. So, it is advised to use it with positive definite matrices.

Value

A list including:

values The eigenvalues.

vectors The eigenvectors.

Author(s)

Armadillo library in C++ and Stefanos Fafalios and Manos Papadakis.

R implementation and documentation: Stefanos Fafalios <stefanosfafalios@gmail.com> and Manos Papadakis <papadakm95@gmail.com>.

See Also

hd.eigen

Examples

## Not run:
x <- matrnorm(500, 100)
s <- Rfast::cova(x)
res<-eigen.sym(s, 5)
x <- s <- NULL
## End(Not run)

Description

Linear models for large scale data.

Usage

lmfit(x, y, w = NULL)
Arguments

x  The design matrix with the data, where each column refers to a different sample of subjects. You must supply the design matrix, with the column of 1s. This function is the analogue of lm.fit and .lm.fit.

y  A numerical vector or a numerical matrix.

w  An optional numerical vector with weights. Note that if you supply this, the function does not make them sum to 1. So, you should do it.

Details

We have simply exploited R’s powerful function and managed to do better than .lm.fit which is a really powerful function as well. This is a bare bones function as it returns only two things, the coefficients and the residuals. .lm.fit returns more and lm.fit even more and finally lm returns too much. The motivation came form this site https://m-clark.github.io/docs/fastr.html. We changed the function a bit.

Value

A list including:

be  The beta coefficients.

residuals  The residuals of the linear model(s).

Author(s)

Michail Tsagris

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

References


See Also

regression, allbetas, correls, mvbetas, cor.fsreg

Examples

n <- 200 ; p <- 5
X <- matrnorm(n, p)
y <- rnorm(n)
a1 <- .lm.fit(X, y)
a2 <- lmfit(X, y)
x <- NULL
Description

Logistic and Poisson regression models.

Usage

```r
glm_logistic(x, y, full = FALSE, tol = 1e-09, maxiters = 100)
glm_poisson(x, y, full = FALSE, tol = 1e-09)
```

Arguments

- `x`: A matrix with the data, where the rows denote the samples (and the two groups) and the columns are the variables. This can be a matrix or a data.frame (with factors).
- `y`: The dependent variable; a numerical vector with two values (0 and 1) for the logistic regression or integer values, 0, 1, 2,... for the Poisson regression.
- `full`: If this is FALSE, the coefficients and the deviance will be returned only. If this is TRUE, more information is returned.
- `tol`: The tolerance value to terminate the Newton-Raphson algorithm.
- `maxiters`: The max number of iterations that can take place in each regression.

Details

The function is written in C++ and this is why it is very fast.

Value

When full is FALSE a list including:

- `be`: The regression coefficients.
- `devi`: The deviance of the model.

When full is TRUE a list including:

- `info`: The regression coefficients, their standard error, their Wald test statistic and their p-value.
- `devi`: The deviance.

Author(s)

Manos Papadakis <papadakm95@gmail.com>

R implementation and documentation: Michail Tsagris <mtsagris@uoc.gr> and Manos Papadakis <papadakm95@gmail.com>.
Logistic or Poisson regression with a single categorical predictor

References


See Also

poisson_only, logistic_only, univglms, regression

Examples

```r
## Not run:
x <- matrix(rnorm(100 * 3), ncol = 3)
y <- rbinom(100, 1, 0.6)  ## binary logistic regression
a1 <- glm_logistic(x, y, full = TRUE)
a2 <- glm(y ~ x, binomial)

x <- matrix(rnorm(100 * 3), ncol = 3)
y <- rpois(100, 10)  ## binary logistic regression
b1 <- glm_poisson(x, y, full = TRUE)
b2 <- glm(y ~ x, poisson)

x<-y<-a1<-a2<-b1<-b2<-NULL
## End(Not run)
```

Description

Logistic or Poisson regression with a single categorical predictor.

Usage

logistic.cat1(y, x, logged = FALSE)
poisson.cat1(y, x, logged = FALSE)

Arguments

y A numerical vector with values 0 or 1.
x A numerical vector with discrete numbers or a factor variable. This is suppose to be a categorical predictor. If you supply a continuous valued vector the function will obviously provide wrong results. **Note:** For the "binomial.anova" if this is a numerical vector it must contain strictly positive numbers, i.e. 1, 2, 3, 4, ..., no zeros are allowed.
logged Should the p-values be returned (FALSE) or their logarithm (TRUE)?
Logistic or Poisson regression with a single categorical predictor

Details

There is a closed form solution for the logistic regression in the case of a single predictor variable. See the references for more information.

Value

info A matrix similar to the one produced by the glm command. The estimates, their standard error, the Wald value and the relevant p-value.

devs For the logistic regression case a vector with the null and the residual deviances, their difference and the significance of this difference.

res For the Poisson regression case a vector with the log likelihood ratio test statistic value and its significance.

Author(s)

Michail Tsagris

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

References


See Also

poisson.anova, poisson.anovas, anova, logistic_only, poisson_only

Examples

```r
y <- rbinom(20000, 1, 0.6)
x <- as.factor( rbinom(20000, 3, 0.5) )
system.time( a1 <- logistic.cat1(y, x) )
system.time( a2 <- glm(y ~ x, binomial) )
a1 ; a2

y <- rpois(20000, 10)
x <- as.factor( rbinom(20000, 3, 0.5) )
system.time( a1 <- poisson.cat1(y, x) )
system.time( a2 <- glm(y ~ x, poisson) )
a1 ; a2

x<-y<-a1<-a2<-NULL
```
Lower and Upper triangular of a matrix

Description

Lower/upper triangular matrix.

Usage

lower_tri(x, suma = FALSE, diag = FALSE)
upper_tri(x, suma = FALSE, diag = FALSE)
lower_tri.assign(x, v, diag = FALSE)
upper_tri.assign(x, v, diag = FALSE)

Arguments

x An matrix with data or a vector with 2 values which is the dimension of the logical matrix to be returned with the upper or lower triangular filled with "TRUE".
v A numeric vector for assign to the lower/upper triangular.
suma A logical value for returning the sum of the upper or lower triangular. By default is "FALSE". Works only if argument "x" is matrix.
diag A logical value include the diagonal to the result.

Value

Get a lower/upper triangular logical matrix with values TRUE/FALSE, a vector with the values of a lower/upper triangular, the sum of the upper/lower triangular if suma is set TRUE or assign to the lower/upper (only for large matrices) triangular. You can also include diagonal with any operation if argument diag is set to "TRUE".

Author(s)

Manos Papadakis
R implementation and documentation: Manos Papadakis <papadakm95@gmail.com>.

See Also

rowMins, colFalse, nth, rowrange, rowMedians, rowVars, colTrue

Examples

x <- matrix(runif(10*10),10,10)
all.equal(lower_tri(c(10,10)),lower.tri(x))
Mahalanobis distance

```
all.equal(lower_tri(x), x[lower.tri(x)])
#all.equal(upper_tri(c(10,10)), upper.tri(x))
#all.equal(upper_tri(x), x[upper.tri(x)])

#all.equal(lower_tri(c(10,10), diag = TRUE), lower.tri(x, diag = TRUE))
#all.equal(lower_tri(x, diag = TRUE), x[lower.tri(x, diag = TRUE)])
#all.equal(upper_tri(c(10,10), diag = TRUE), upper.tri(x, diag = TRUE))
#all.equal(upper_tri(x, diag = TRUE), x[upper.tri(x, diag = TRUE)])
all.equal(lower_tri.assign(x, diag = TRUE, v=rep(1,1000)), x[lower.tri(x, diag = TRUE)]<-1)
all.equal(upper_tri.assign(x, diag = TRUE, v=rep(1,1000)), x[upper.tri(x, diag = TRUE)]<-1)
x<-NULL
```

Description

Mahalanobis distance.

Usage

```
mahala(x, mu, sigma, ischol = FALSE)
```

Arguments

- **x**: A matrix with the data, where rows denotes observations (vectors) and the columns contain the variables.
- **mu**: The mean vector.
- **sigma**: The covariance or any square symmetric matrix.
- **ischol**: A boolean variable set to true if the Cholesky decomposition of the covariance matrix is supplied in the argument `sigma`.

Value

A vector with the Mahalanobis distances.
Many (and one) area under the curve values

Author(s)
Matteo Fasiolo <matteo.fasiolo@gmail.com>,
C++ and R implementation and documentation: Matteo Fasiolo <matteo.fasiolo@gmail.com>.

See Also
dista, colmeans

Examples

```r
x <- matrix( rnorm(100 * 50), ncol = 50 )
m <- colmeans(x)
s <- cov(x)
a1 <- mahala(x, m, s)
```

Description

Many are under the curve values.

Usage

```r
colaucs(group, preds)
auc(group, preds)
```

Arguments

- `group`: A numerical vector with two values, one of which must be strictly 1.
- `preds`: A numerical matrix with scores, probabilities or any other measure. In the case of auc this is a vector.

Details

The AUCs are calculated column-wise or just an AUC if the vector function is used.

Value

A vector with length equal to the number of columns of the "preds" argument. The AUC values for each column. If the "auc" function is used then a single number is returned.

Author(s)
Michail Tsagris
R implementation and documentation: Michail Tsagris <mtsagris@uoc.gr> and Manos Papadakis <papadakm95@gmail.com>.
Many 2 sample proportions tests

See Also
tests, ttest, ftests

Examples

```r
## 200 variables, hence 200 AUCs will be calculated
x <- matrix(rnorm(100 * 200), ncol = 200)
ina <- rbinom(100, 1, 0.6)
system.time(colaucs(ina, x))
a <- colaucs(ina, x)
b <- auc(ina, x[, 1])
x <- NULL
```

Description

It performs very many 2 sample proportions tests.

Usage

```r
proptests(x1, x2, n1, n2)
```

Arguments

- `x1`: A vector with the successes of the one group.
- `x2`: A vector with the successes of the one group.
- `n1`: A vector with the number of trials of the one group.
- `n2`: A vector with the number of trials of the one group.

Details

The 2-sample proportions test is performed for each pair of proportions of the two groups.

Value

A matrix with the proportions of each group (two columns), the test statistic and the p-value of each test.

Author(s)

Michail Tsagris

R implementation and documentation: Michail Tsagris <mtsagris@uoc.gr> and Manos Papadakis <papadakm95@gmail.com>.
Many 2 sample tests

References

See Also
ttests, ftests, colVars

Examples

```r
## 10000 variables, hence 10000 t-tests will be performed
set.seed(12345)
x1 <- rpois(500, 5)
x2 <- rpois(500, 5)
n1 <- rpois(1000, 40)
n2 <- rpois(1000, 40)
a <- proptests(x1, x2, n1, n2)
mean(a[, 4]<0.05)

x1 <- rbinom(500, 500, 0.6)
x2 <- rbinom(500, 500, 0.6)
b <- proptests(x1, x2, 500, 500)
mean(b[, 4]<0.05)
```

Description
It performs very many 2 sample tests.

Usage
ttests(x, y = NULL, ina, paired = FALSE, logged = FALSE, parallel = FALSE)
mcnemars(x, y = NULL, ina, logged = FALSE)
var2tests(x, y = NULL, ina, alternative = "unequal", logged = FALSE)

Arguments
- x: A matrix with the data, where the rows denote the samples and the columns are the variables.
- y: A second matrix with the data of the second group. If this is NULL (default value) then the argument ina must be supplied. Notice that when you supply the two matrices the procedure is two times faster.
- ina: A numerical vector with 1s and 2s indicating the two groups. Be careful, the function is designed to accept only these two numbers. In addition, if your "y" is NULL, you must specify "ina".
Many 2 sample tests

alternative  The type of hypothesis to be checked, "equal", "greater", "less".

paired     If the groups are not independent paired t-tests should be performed and this must be TRUE, otherwise, leave it FALSE. In this case, the two groups must have equal sample sizes, otherwise no test will be performed.

logged     Should the p-values be returned (FALSE) or their logarithm (TRUE)?

parallel   Should parallel implementations take place in C++? The default value is FALSE.

Details

For the ttests, if the groups are independent, the Welch’s t-test (without assuming equal variances) is performed. Otherwise many paired t-tests are performed. The McNemar’s test requires a number of observations, at least 30 would be good in order for the test to have some power and be size corect.

Value

A matrix with the test statistic, the degrees of freedom (if the groups are independent) and the p-value (or their logarithm) of each test.

Author(s)

Michail Tsagris

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

References


See Also

ftests, anovas, ttest

Examples

```r
## 1000 variables, hence 1000 t-tests will be performed
x = matrnorm(100, 100)
## 100 observations in total
ina = rbinom(100, 1, 0.6) + 1  ## independent samples t-test
system.time( ttests(x, ina = ina) )
x1 = x[ina == 1, ]
x2 = x[ina == 2, ]
system.time( ttests(x1, x2) )
x <- NULL
```
Many analysis of variance tests with a discrete variable

Description

Many analysis of variance tests with a discrete variable.

Usage

```r
poisson.anovas(y, ina, logged = FALSE)
quasipoisson.anovas(y, ina, logged = FALSE)
gem.anovas(y, ina, type = 1, logged = FALSE)
```

Arguments

- `y`: A numerical matrix with discrete valued data, i.e. counts for the case of the Poisson, or with 0s and 1s for the case of the Bernoulli distribution. Each column represents a variable.
- `ina`: A numerical vector with discrete numbers starting from 1, i.e. 1, 2, 3, 4,... or a factor variable. This is suppose to be a categorical predictor. If you supply a continuous valued vector the function will obviously provide wrong results.
- `type`: This argument is for the geometric distribution. Type 1 refers to the case where the minimum is zero and type 2 for the case of the minimum being 1.
- `logged`: Should the p-values be returned (FALSE) or their logarithm (TRUE)?

Details

This is the analysis of variance with count data. What we do is many log-likelihood ratio tests. For the quasi Poisson case we scale the difference in the deviances.

Value

A matrix with two values, the difference in the deviances (test statistic) and the relevant p-value. For the case of quasi Poisson the estimated $\phi$ parameter is also returned.

Author(s)

Michail Tsagris

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

See Also

`g2tests,poisson.anova,anova,poisson_only,poisson.mle`
Examples

```r
ina <- rbinom(500, 3, 0.5) + 1
## Poisson example
y <- matrix( rpois(500 * 100, 10), ncol= 100 )
system.time(a1 <- poisson.anovas(y, ina) )
y <- NULL
```

Description

Many ANCOVAs.

Usage

```r
ancovas(y, ina, x, logged = FALSE)
```

Arguments

- `y` A matrix with the data, where the rows denote the observations and the columns are the variables.
- `ina` A numerical vector with 1s, 2s, 3s and so one indicating the two groups. Be careful, the function is designed to accept numbers greater than zero.
- `x` A numerical vector whose length is equal to the number of rows of `y`. This is the covariate.
- `logged` Should the p-values be returned (FALSE) or their logarithm (TRUE)?

Details

Many Analysis of covariance tests are performed. No interaction between the factor and the covariate is tested. Only the main effects. The design need not be balanced. The values of `ina` need not have the same frequency. The sums of squares have been adjusted to accept balanced and unbalanced designs.

Value

A matrix with the test statistic and the p-value for the factor variable and the covariate.

Author(s)

Michail Tsagris

R implementation and documentation: Michail Tsagris <mtsagris@uoc.gr> and Manos Papadakis <papadakm95@gmail.com>.
**Many ANOVAS for count data with Poisson or quasi Poisson models**

**References**


**See Also**

`ftests, ttests, anovas`

**Examples**

```r
## 100 variables, hence 100 F-tests will be performed
y <- matrix(rnorm(90 * 100), ncol = 100)
ina <- rbinom(90, 2, 0.5) + 1
x <- rnorm(90)

system.time(a <- ancovas(y, ina, x))

## Not run:
m1 <- lm(y[, 15] ~ factor(ina) + x)
m2 <- lm(y[, 15] ~ x + factor(ina))
res<anova(m1)
res<anova(m2)
y <- NULL

a[15, ] ## the same with the m2 model, but not the m1

## End(Not run)
```

**Description**

Many ANOVAS for count data with Poisson or quasi Poisson models.

**Usage**

```r
colpoisson.anovas(y, x, logged = FALSE)
colquasipoisson.anovas(y, x, logged = FALSE)
```

**Arguments**

- `y`: A numerical vector with the data.
- `x`: A matrix with the data, where the rows denote the samples (and the two groups) and the columns are the variables. This must be a matrix with the categorical variables as numbers, starting from 1. Poisson or quasi Poisson ANOVA takes place for each column.
- `logged`: A boolean variable; it will return the logarithm of the pvalue if set to TRUE.
Many exponential regressions

Details

Poisson or quasi Poisson ANOVA takes place at each column.

Value

A matrix with the test statistic and the (logged) p-value for each predictor variable. In the case of the quasi Poisson, the $\phi$ is returned as well.

Author(s)

Michail Tsagris and Manos Papadakis

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

See Also

poisson.anova boot.ttest2, ttest2, ftest

Examples

```r
y <- rpois(200, 10)
x <- matrix(rbinom(200 * 10, 3, 0.5 ), ncol = 10)
```
Many F-tests with really huge matrices

Details
We have implemented the newton-Raphson in order to avoid unnecessary calculations.

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
univglmss, score.glms, logistic_only, poisson_only, regression

Examples
## 200 variables, hence 200 univariate regressions are to be fitted
x <- matrnorm(100, 100)
y <- rexp(100, 4)
system.time( expregs(y, x, di = rep(1, length(y))) )
x <- NULL

Many F-tests with really huge matrices

Description
Many F-tests with really huge matrices.

Usage
list.ftests(x, logged = FALSE)

Arguments
x A list with many big size matrices. Each element of the list contains a matrix. This is the ftests function but with really huge matrices, which cannot be loaded into R as a single matrix.

logged Should the p-values be returned (FALSE) or their logarithm (TRUE)?
Details

The Welch's F-test (without assuming equal variances) is performed just like in the "ftests" function. The difference is that you have a really huge matrix which you cannot load into R. In the "ftests" function, the argument "ina" denotes the different groups. Here, you "cut" the matrix into smaller ones, each of which denotes a different group and put them in a list.

Value

A matrix with the test statistic and the p-value of each test.

Author(s)

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

References


See Also

`ftests, ttests`

Examples

```r
x <- matrnorm(300, 500)
ina <- rbinom(300, 2, 0.6) + 1
a <- list()
a[[ 1 ]] <- x[ina == 1 , ]
a[[ 2 ]] <- x[ina == 2 , ]
a[[ 3 ]] <- x[ina == 3 , ]
mod <- list.ftests(a)
z <- NULL
a <- NULL
```

Many G-square and Chi-square tests of independence

Many G-square and Chi-square tests of independence

Description

Many G-square tests of independence with and without permutations.

Usage

```r
g2tests(data, x, y, dc)
g2tests_perm(data, x, y, dc, nperm)
chi2tests(data, x, y, dc)
```
Many G-square and Chi-square tests of independence

Arguments

- **data**: A numerical matrix with the data. **The minimum must be 0, otherwise the function can crash or will produce wrong results.** The data must be consecutive numbers.
- **x**: An integer number or a vector of integer numbers showing the other variable(s) to be used for the $G^2$ test of independence.
- **y**: An integer number showing which column of data to be used.
- **dc**: A numerical value equal to the number of variables (or columns of the data matrix) indicating the number of distinct, unique values (or levels) of each variable. Make sure you give the correct numbers here, otherwise the degrees of freedom will be wrong.
- **nperm**: The number of permutations. The permutations test is slower than without permutations and should be used with small sample sizes or when the contingency tables have zeros. When there are few variables, R’s "chisq.test" function is faster, but as the number of variables increase the time difference with R’s procedure becomes larger and larger.

Details

The function does all the pairwise $G^2$ test of independence and gives the position inside the matrix. The user must build the associations matrix now, similarly to the correlation matrix. See the examples of how to do that. The p-value is not returned, we leave this to the user. See the examples of how to obtain it.

Value

A list including:

- **statistic**: The $G^2$ or $\chi^2$ test statistic for each pair of variables.
- **pvalue**: This is returned when you have selected the permutation based $G^2$ test.
- **x**: The row or variable of the data.
- **y**: The column or variable of the data.
- **df**: The degrees of freedom of each test.

Author(s)

Giorgos Borboudakis. The permutation version used a C++ code by John Burkardt.

R implementation and documentation: Manos Papadakis <papadakm95@gmail.com>.

References


Many Gini coefficients

See Also

g2Test, g2Test_perm, correls, univglm

Examples

```r
nvalues <- 3
nvars <- 10
nsamples <- 2000
data <- matrix( sample( 0:(nvalues - 1), nvars * nsamples, replace = TRUE ), nsamples, nvars )
dc <- rep(nvalues, nvars)
a <- g2tests(data = data, x = 2:9, y = 1, dc = dc)
pval <- pchisq(a$statistic, a$df, lower.tail = FALSE) ## p-value
b <- g2tests_perm(data = data, x = 2:9, y = 1, dc = dc, nperm = 1000)
a <- b <- data <- NULL
```

Many Gini coefficients

Description

Many Gini coefficients.

Usage

```r
ginis(x)
```

Arguments

- `x` A matrix with non negative data. The rows are observations and the columns denote the variables.

Details

We have implemented the fast version of the Gini coefficient. See wikipedia for more details.

Value

A vector with the Gini coefficient, one for each variable.

Author(s)

Michail Tsagris

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

See Also

colskewness, colmeans, corpairs
Many hypothesis tests for two means of percentages

Examples

```r
x <- matrix( rpois(500 * 1000, 1000), ncol = 1000 )
a <- ginis(x)
```

Description

Many hypothesis tests for two means of percentages.

Usage

```r
percent.ttests(x, y, logged = FALSE)
```

Arguments

- `x`: A numerical matrix with the percentages of the first sample. Any value between 0 and 1 (inclusive) is allowed.
- `y`: A numerical matrix with the percentages of the first sample. Any value between 0 and 1 (inclusive) is allowed.
- `logged`: Should the p-values be returned (FALSE) or their logarithm (TRUE)?

Details

This is the `prop.reg` but with a single categorical predictor which has two levels only. It is like a t-test for the means of two samples having percentages.

Value

A matrix with three columns, the phi parameter, the test statistic and its associated p-value.

Author(s)

Michail Tsagris

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

References


Many moment and maximum likelihood estimations of variance components

See Also

link{percent.ttest}, \texttt{prop.reg}, \texttt{ttest2}, \texttt{ftest}

Examples

\begin{verbatim}
x <- matrix( rbeta(100 * 10, 3, 1), ncol = 10)
y <- matrix( rbeta(50 * 10, 7.5, 2.5), ncol = 10)
res<--percent.ttests(x, y)
\end{verbatim}

Description

Many moment and maximum likelihood estimations of variance components.

Usage

\begin{verbatim}
colvarcomps.mom(x, id, parallel = FALSE)
colvarcomps.mle(x, id, ranef = FALSE, tol= 1e-08, maxiters = 100, parallel = FALSE)
\end{verbatim}

Arguments

\begin{verbatim}
x \hspace{1cm} A matrix with the data, where each column refers to a different sample of subjects.
id \hspace{1cm} A numerical vector indicating the subject. You must put consecutive numbers and no zero values. Alternatively this can be a factor variable.
ranef \hspace{1cm} Do you also want the random effects to be returned? TRUE or FALSE.
tol \hspace{1cm} The tolerance level to terminate the golden ratio search.
maxiters \hspace{1cm} The maximum number of iterations to perform.
parallel \hspace{1cm} Should the computations run in parallel? TRUE or FALSE.
\end{verbatim}

Details

Note that the "colvarcomp.mom" works for \textbf{balanced designs only}, i.e. for each subject the same number of measurements have been taken. The "colvarcomps.mle" works for unbalanced as well.

The variance components, the variance of the between measurements and the variance of the within are estimated using moment estimators. The "colvarcomps.mom" is the moment analogue of a random effects model which uses likelihood estimation ("colvarcomps.mle"). It is much faster, but can give negative variance of the random effects, in which case it becomes zero.

The maximum likelihood version is a bit slower (try yourselves to see the difference), but statistically speaking is to be preferred when small samples are available. The reason why it is only a little
Many moment and maximum likelihood estimations of variance components

bit slower and not a lot slower as one would imagine is because we are using a closed formula to calculate the two variance components (Demidenko, 2013, pg. 67-69). Yes, there are closed formulas for linear mixed models.

**Value**

For the "colvarcomps.mom": A matrix with 5 columns, The MSE, the estimate of the between variance, the variance components ratio and a 95% confidence for the ratio.

For the "colvarcomps.mle": If ranef = FALSE a list with a single component called "info". That is a matrix with 3 columns, The MSE, the estimate of the between variance and the log-likelihood value. If ranef = TRUE a list including "info" and an extra component called "ranef" containing the random effects. It is a matrix with the same number of columns as the data. Each column contains the random effects of each variable.

**Author(s)**

Michail Tsagris

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

**References**


**See Also**

varcomps.mle, colrint.regbx

**Examples**

```r
## example taken from Montgomery, page 514-517.
y <- c(98, 97, 99, 96, 91, 90, 93, 92, 96, 95, 97, 95, 95, 96, 99, 98)
y <- matrix(y)
id <- rep(1:4, each = 4)
x <- rmvnorm(100, numeric(100), diag(rexp(100)))

x <- c(98, 97, 99, 96, 91, 90, 93, 92, 96, 95, 97, 95, 95, 96, 99, 98)
y <- matrix(y)
id <- rep(1:4, each = 4)
x <- NULL
```

```r
mean(a[, 4]<0 & a[, 5]>0)
```
Many multi-sample tests

Description

Many multi-sample tests.

Usage

ftests(x, ina, logged = FALSE)
anovas(x, ina, logged = FALSE)
vartests(x, ina, type = "levene", logged = FALSE)
block.anovas(x, treat, block, logged = FALSE)

Arguments

x        A matrix with the data, where the rows denote the observations (and the two
groups) and the columns are the variables.

ina      A numerical vector with 1s, 2s, 3s and so one indicating the two groups. Be care-
         ful, the function is designed to accept numbers greater than zero. Alternatively
         it can be a factor variable.

type     This is for the variances test and can be either "levene" or "bf" corresponding to
         Levene's or Brown-Forsythe's testing procedure.

treat    In the case of the blocking ANOVA this argument plays the role of the "ina"
         argument.

block    This item, in the blocking ANOVA denotes the subjects which are the same.
         Similarly to "ina" a numeric vector with 1s, 2s, 3s and so on.

logged   Should the p-values be returned (FALSE) or their logarithm (TRUE)?

Details

The Welch’s F-test (without assuming equal variances) is performed with the "ftests" function. The
"anovas" function perform the classical (Fisher's) one-way analysis of variance (ANOVA) which
assumes equal variance across the groups.

The "vartests" perform hypothesis test for the equality of the variances in two ways, either via
the Levene or via the Brown-Forsythe procedure. Levene’s test employs the means, whereas
the Brown-Forsythe procedure employs the medians and is therefore more robust to outliers. The
"var2tests" implement the classical F test.

The "block.anova" is the ANOVA with blocking, randomised complete block design (RCBD). In
this case, for every combination of the block and treatment values, there is only one observation.
The mathematics are the same as in the case of two way ANOVA, but the assumptions different and
the testing procedure also different. In addition, no interaction is present.
**Value**

A matrix with the test statistic and the p-value of each test.

**Author(s)**

Michail Tsagris

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

**References**


**See Also**

tests

**Examples**

```r
x <- matrix( rnorm(300 * 50), ncol = 50 )
## 300 observations in total
ina <- rbinom(300, 3, 0.6) + 1
a1 <- ftests(x, ina)
a2 <- anovas(x, ina)
a3 <- vartests(x, ina)
x <- NULL
```

**Description**

Many multivariate simple linear regressions coefficients.

**Usage**

`mvbetas(y, x, pvalue = FALSE)`

**Arguments**

- `y`: A matrix with the data, where rows denotes the observations and the columns contain the dependent variables.
- `x`: A numerical vector with one continuous independent variable only.
- `pvalue`: If you want a hypothesis test that each slope (beta coefficient) is equal to zero set this equal to TRUE. It will also produce all the correlations between `y` and `x`. 
Many non parametric multi-sample tests

Details

It is a function somehow opposite to the `allbetas`. Instead of having one `y` and many `xs` we have many `ys` and one `x`.

Value

A matrix with the constant (`alpha`) and the slope (`beta`) for each simple linear regression. If the `p-value` is set to `TRUE`, the correlation of each `y` with the `x` is calculated along with the relevant `p-value`.

Author(s)

Michail Tsagris

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

See Also

`allbetas`, `correls`, `univglm`

Examples

```r
y <- matrnorm(100, 100)
x <- rnorm(100)
a <- mvbetas(y, x, pvalue = FALSE)
b <- matrix(nrow = 100, ncol = 2)
z <- cbind(1, x)

system.time( a <- mvbetas(y, x) )
b[2, ] <- coef( lm.fit( z, y[, 1] ) )
b[2, ] <- coef( lm.fit( z, y[, 2] ) )
x <- NULL
```
Arguments

x  
A matrix with the data, where the rows denote the samples (and the two groups) 
and the columns are the variables.

ina  
A numerical vector with 1s, 2s, 3s and so one indicating the two groups. Be 
careful, the function is designed to accept numbers greater than zero.

treat  
In the case of the Cochran’s Q test, this argument plays the role of the “ina” 
argument.

block  
This item denotes the subjects which are the same. Similarly to “ina” a numeric 
vector with 1s, 2s, 3s and so on.

logged  
Should the p-values be returned (FALSE) or their logarithm (TRUE)?

Details

The "kruskaltests" performs the Kruskal-Wallis non parametric alternative to analysis of variance 
test. The "cqtests" performs the Cochran’s Q test for the equality of more than two groups whose 
values are strictly binary (0 or 1). This is a generalisation of the McNemar’s test in the multi-sample 
case.

Value

A matrix with the test statistic and the p-value of each test.

Author(s)

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

See Also

block.anovas, ftests

Examples

```r
x <- matrix( rexp(300 * 200), ncol = 200 )
ina <- rbinom(300, 3, 0.6) + 1
system.time( kruskaltests(x, ina) )
x <- matrix( rbinom(300 * 200, 1, 0.6), ncol = 200 )
treat <- rep(1:3, each = 100)
block <- rep(1:3, 100)
system.time( cqtests(x, treat, block) )
x <- NULL
```
Many odds ratio tests

Description

It performs very many odds ratio tests.

Usage

odds(x, y = NULL, ina, logged = FALSE)

Arguments

x
A matrix with the data, where the rows denote the observations and the columns are the variables. They must be 0s and 1s only.

y
A second matrix with the data of the second group. If this is NULL (default value) then the argument ina must be supplied. Notice that when you supply the two matrices the procedure is two times faster. They must be 0s and 1s only.

ina
A numerical vector with 1s and 2s indicating the two groups. Be careful, the function is designed to accept only these two numbers. In addition, if your "y" is NULL, you must specify "ina".

logged
Should the p-values be returned (FALSE) or their logarithm (TRUE)?

Details

Many odds ratio tests are performed.

Value

A matrix with the test statistic and the p-value (or their logarithm) of each test.

Author(s)

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

References


See Also

odds.ratio, g2Test_univariate
Many one sample goodness of fit tests for categorical data

Examples

```r
x <- matrix( rbinom(100 * 100, 1, 0.5), ncol = 100 )
ina <- rep(1:2, each = 50)
a <- odds(x, ina = ina)
```

Description

Many one sample goodness of fit tests for categorical data.

Usage

```r
cat.goftests(x, props, type = "gsquare", logged = FALSE)
```

Arguments

- `x`: A matrix with the data, where the rows denote the samples and the columns are the variables. The data must be integers and be of the form 1, 2, 3, and so on. The minimum must be 1, and not zero.
- `props`: The assumed distribution of the data. A vector or percentages summing to 1.
- `type`: Either Pearson’s $\chi^2$ test (“chisquare”) is used or the $G^2$ test (“qsquare”, default value).
- `logged`: Should the p-values be returned (FALSE) or their logarithm (TRUE)?

Details

Given a matrix of integers, where each column refers to a sample, the values of a categorical variable the function tests whether these values can be assumed to fit a specific distribution.

Value

A matrix with the test statistic and the p-value of each test.

Author(s)

Michail Tsagris

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

See Also

`ttests`, `ttest`, `ftests`
Many one sample tests

Examples

```r
x <- matrix(rbinom(300 * 100, 4, 0.6), ncol = 100) + 1
props <- dbinom(0:4, 4, 0.6)
## can we assume that each column comes from a distribution whose mass is given by props?
system.time(cat.goftests(x, props))
a1 <- cat.goftests(x, props)  ## G-square test
a2 <- cat.goftests(x, props, type = "chisq")  ## Chi-square test
cor(a1, a2)
mean(abs(a1 - a2))
x <- NULL
```

Description

Many one sample tests.

Usage

```r
proptest(x, n, p, alternative = "unequal", logged = FALSE)
ttest(x, m, alternative = "unequal", logged = FALSE, conf = NULL)
vartest(x, sigma, alternative = "unequal", logged = FALSE, conf = NULL)
```

Arguments

- `x`: A matrix with numerical data. Each column of the matrix corresponds to a sample, or a group. In the case of the "proptest" this is a vector integers ranging from 0 up to n. It is the number of "successes".
- `n`: This is for the "proptest" only and is a vector with integer numbers specifying the number of tries for the proptest. Its size is equal to the size of x.
- `p`: A vector with the assumed probabilities of success in the "proptest". Its size is equal to the number of columns of the matrix x.
- `m`: A vector with the assumed means. Its size is equal to the number of columns of the matrix x.
- `sigma`: A vector with assumed variances. Its size is equal to the number of columns of the matrix x.
- `alternative`: The type of hypothesis to be checked. Equal to ("unequal"), greater than("greater") or less than ("less") the assumed parameter.
- `logged`: Should the p-values be returned (FALSE) or their logarithm (TRUE)?
- `conf`: If you want confidence intervals to be returned specify the confidence level, otherwise leave it NULL.

Details

Despite the functions having been written in R, they are very fast.
Many random intercepts LMMs for balanced data with a single identical covariate.

Value

For all tests except for the "sftests" a matrix with two columns, the test statistic and the p-value respectively.

Author(s)

Michail Tsagris

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

See Also

\texttt{ftests}, \texttt{ttests}

Examples

\begin{verbatim}
R <- 100
## protest
x <- rbinom(R, 50, 0.6)
n <- rep(50, R)
p <- rep(0.6, R)
a1 <- proptest(x, n, p, "unequal", logged = FALSE)
res<-sum( a1[, 2] < 0.05 ) / R

## vartest
x <- matrnorm(100, 100)
a2 <- vartest(x, rep(1, R) )
res<-sum( a2[, 2] < 0.05 )

## ttest
a4 <- ttest(x, numeric(R) )
res<-sum(a4[, 2] < 0.05 ) / R
x <- NULL
\end{verbatim}
Many random intercepts LMMs for balanced data with a single identical covariate.

Arguments

y  A numerical matrix with the data. The subject values.

x  A numerical vector with the same length as the number of rows of y indicating the fixed predictor variable. Its values are the same for all levels of y. An example of this x is time which is the same for all subjects.

id  A numerical variable with 1, 2, ... indicating the subject.

Details

This is a special case of a balanced random intercepts model with a compound symmetric covariance matrix and one single covariate which is constant for all replicates. An example, is time, which is the same for all subjects. Maximum likelihood estimation has been performed. In this case the mathematics exist in a closed formula (Demidenko, 2013, pg. 67-69).

This is the generalistion of rint.regbx to matrices. Assume you have many observations, gene expressions over time for example, and you want to calculate the random effects or something else for each expression. Instead of using a "for" loop with rint.regbx function we have used amtrix operations to make it even faster.

Value

A list including:

info  A matrix with the random intercepts variance (between), the variance of the errors (within), the log-likelihood, the deviance (twice the log-likelihood) and the BIC. In the case of "rint.reg" it also includes the number of iterations required by the generalised least squares.

be  The estimated regression coefficients, which in the case of "rint.regbx" are simply two: the constant and the slope (time effect).

ranef  A matrix with random intercepts effects. Each row corresponds to a column in y. Instead of having a matrix with the same number of columns as y we return a transposed matrix.

Author(s)

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

References


See Also

colvarcomps.mle, rint.regbx, rm.lines, varcomps.mom, rint.reg
Examples

```r
y <- matrix( rnorm(100 * 50), ncol = 50)
id <- rep(1:20, each = 5)
x <- rep(1:10, 10)
-system.time( a<- colrint.regbx(y, x, id) )
```

Description

Many regression based tests for single sample repeated measures.

Usage

```r
rm.lines(y, x, logged = FALSE)
rm.anovas(y, x, logged = FALSE)
```

Arguments

- `y`: A matrix with the data, where each column refers to a different sample of subjects. For example, the first column is the repeated measurements of a sample of subjects, the second column contains repeated measurements of a second sample of subjects and so on. Within each column, the measurements of each subjects are stacked one upon the other. Say for example there are n subjects and each of them has been measured d times (in time or at different experimental conditions). We put these in a matrix with just one column. The first d rows are the measurements of subject 1, the next d rows are the measurements of subject 2 and so on.

- `x`: A numerical vector with time (usually) or the predictor variable. For example the temperature, or the pressure. See the details for more information. Its length is equal to the time points for example, i.e. it must not have the same length as the number of rows of `y`. For the "rm.lines" this is a continuous variable. For the "rm.anovas" this is treated as a categorical variable, indicating say the type of experimental condition, but no difference between the points is important. Hence, for this function only, `x` can also be a facto variable.

- `logged`: Should the p-values be returned (FALSE) or their logarithm (TRUE)?

Details

In order to see whether the repeated measurements are associated with a single covariate, e.g. time we perform many regressions and each time calculate the slope. For each subject, its regression slope with the covariate is calculated. In the end a t-test for the hypothesis that the average slopes is zero is performed. The regression slopes ignore that the measurements are not independent, but
Many regression based tests for single sample repeated measures

note that the slopes are independent, because they come from different subjects. This is a simple, summary statistics based approach found in Davis (2002), yet it can provide satisfactory results. The second approach ("rm.anovas") found in Davis (2002) is the usual repeated measures ANOVA. In this case, suppose you have taken measurements on one or more variables from the same group of people. See the example below on how to put such data.

Value

A matrix with the test statistic (t-test) and its associated p-value.

Author(s)

Michail Tsagris

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

References


See Also

rint.regbx, rint.reg, varcomps.mle

Examples

y <- c(74.5, 81.5, 83.6, 68.6, 73.1, 79.4,
      75.5, 84.6, 70.6, 87.3, 73.0, 75.0,
      68.9, 71.6, 55.9, 61.9, 60.5, 61.8,
      57.0, 61.3, 54.1, 59.2, 56.5, 58.8,
      78.3, 84.9, 64.0, 62.2, 60.1, 78.7,
      54.0, 62.8, 63.0, 58.0, 56.0, 51.5,
      72.5, 68.3, 67.8, 71.5, 65.0, 67.7,
      80.8, 89.9, 83.2, 83.0, 85.7, 79.6)
y <- as.matrix(y)
### the first 6 measurements are from subject 1, measurements 7-12 are from subject 2,
# measurements 13-18 are from subject 3 and so on.
x <- c(-10, 25, 37, 50, 65, 80) ## all subjects were measured at the same time points
res<-rm.lines(y, x) ## Is linear trend between the measurements and the temperature?
res<-rm.anovas(y, x) ## Tests whether the means of the individuals are the same
## the temperature is treated as categorical variable here.

## fake example
y <- matrnorm(10, 4)
## the y matrix contains 4 repeated measurements for each of the 10 persons.
x <- 1:4
## we stack the measurements of each subject, one under the other in a matrix form.
y1 <- matrix(t(y))
res<-rm.anovas(y1, x) ## perform the test
z <- matrix(rnorm(20 * 8), ncol = 2) ## same example, but with 2 sets of measurements.
Many score based regressions

res<-rm.anovas(z, x)

Many score based regressions

Description

Many score based GLM regressions.

Usage

score.glms(y, x, oiko = NULL, logged = FALSE)
score.multinomregs(y, x, logged = FALSE)
score.negbinregs(y, x, type = 1, logged = FALSE)
score.weibregs(y, x, logged = FALSE)
score.betaregs(y, x, logged = FALSE)
score.gammaregs(y, x, logged = FALSE)
score.expregs(y, x, logged = FALSE)
score.invgaussregs(y, x, logged = FALSE)
score.ztpregs(y, x, logged = FALSE)
score.geomregs(y, x, logged = FALSE)

Arguments

y A vector with either discrete or binary data for the Poisson, geometric, or negative binomial and binary logistic regressions, respectively. A vector with discrete values or factor values for the multinomial regression. If the vector is binary and choose multinomial regression the function checks and transfers to the binary logistic regression.

For the Weibull, gamma, inverse Gaussian and exponential regressions they must be strictly positive data, lifetimes or durations for example. For the beta regression they must be numbers between 0 and 1. For the zero truncated Poisson regression (score.ztpregs) they must be integer valued data strictly greater than 0.

x A matrix with data, the predictor variables.

oiko This can be either "poisson" or "binomial". If you are not sure leave it NULL and the function will check internally.

type This argument is for the negative binomial distribution. In the negative binomial you can choose which way your prefer. Type 1 is for small sample sizes, whereas type 2 is for larger ones as is faster.

logged A boolean variable; it will return the logarithm of the p-value if set to TRUE.
Many score based regressions

Details

Instead of maximising the log-likelihood via the Newton-Raphson algorithm in order to perform the hypothesis testing that $\beta_i = 0$ we use the score test. This is dramatically faster as no model needs to be fitted. The first derivative (score) of the log-likelihood is known and in closed form and under the null hypothesis the fitted values are all equal to the mean of the response variable $y$. The variance of the score is also known in closed form. The test is not the same as the likelihood ratio test. It is size correct nonetheless but it is a bit less efficient and less powerful. For big sample sizes though (5000 or more) the results are the same. We have seen via simulation studies is that it is size correct to large sample sizes, at last a few thousands. You can try for yourselves and see that even with 500 the results are pretty close. The score test is pretty faster than the classical log-likelihood ratio test.

Value

A matrix with two columns, the test statistic and its associated p-value. For the Poisson and logistic regression the p-value is derived via the t distribution, whereas for the multinomial regressions via the $\chi^2$ distribution.

Author(s)

Michail Tsagris.

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

References


See Also

univglms, logistic_only, poisson_only, regression

Examples

```r
x <- matrnorm(500, 500)
y <- rbinom(500, 1, 0.6)  ## binary logistic regression
a2 <- score.glms(y, x)
```
```
y <- rweibull(500, 2, 3)
a <- score.weibregs(y, x)
mean(a[, 2] < 0.05)
x <- NULL
```

---

### Many Shapiro-Francia normality tests

**Description**

Many Shapiro-Francia normality tests.

**Usage**

```r
sftests(x, logged = FALSE)
sftest(x, logged = FALSE)
```

**Arguments**

- **x**: A matrix with the data, where the rows denote the observations and the columns are the variables. In the case of a single sample, then this must be a vector and "sftest" is to be used.
- **logged**: Should the p-values be returned (FALSE) or their logarithm (TRUE)?

**Details**

The Shapiro-Francia univariate normality test is performed for each column (variable) of the matrix `x`.

**Value**

A matrix with the squared correlation between the ordered values and the standard normal ordered statistics, the test statistic and the p-value of each test. If the "sftest" has been used, the output is a vector with these three elements.

**Author(s)**

Michail Tsagris

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

**References**


Many simple circular or angular regressions

See Also
ttests, ttest, ftests

Examples

```r
x <- matrnorm(200, 100)
system.time( sfests(x) )
a <- sfests(x)
mean(a[, 3]<0.05)
x <- rnorm(100)
res<-sftest(x)
```

Many simple circular or angular regressions

Many simple circular or angular regressions

Description

Many regressions with one circular dependent variable and one Euclidean independent variable.

Usage

```r
spml.regs(y, x, tol = 1e-07, logged = FALSE, maxiters = 100, parallel = FALSE)
```

Arguments

- `y`: The dependent variable, it can be a numerical vector with data expressed in radians or it can be a matrix with two columns, the cosinus and the sinus of the circular data. The benefit of the matrix is that if the function is to be called multiple times with the same response, there is no need to transform the vector every time into a matrix.
- `x`: A matrix with independent variable.
- `tol`: The tolerance value to terminate the Newton-Raphson algorithm.
- `logged`: Do you want the logarithm of the p-value be returned? TRUE or FALSE.
- `maxiters`: The maximum number of iterations to implement.
- `parallel`: Do you want the calculations to take place in parallel? The default value if FALSE.

Details

The Newton-Raphson algorithm is fitted in these regression as described in Presnell et al. (1998). For each column of `x` a circual regression model is fitted and the hypothesis testing of no association between `y` and this variable is performed.

Value

A matrix with two columns, the test statistics and their associated (log) p-values.
Many simple geometric regressions

Author(s)
Michail Tsagris and Stefanos Fafalios
R implementation and documentation: Michail Tsagris <mtsagris@uoc.gr> and Stefanos Fafalios <stefanosfafalios@gmail.com>

References

See Also
spml.mle, iag.mle, acg.mle

Examples
```r
x <- rnorm(100)
z <- cbind(3 + 2 * x, 1 - 3 * x)
y <- cbind(rnorm(100, z[,1], 1), rnorm(100, z[,2], 1))
y <- y / sqrt(rowsums(y^2))
x <- matrnorm(100, 100)
a <- spml.regs(y, x)
x <- NULL
```

Description
Many simple geometric regressions.

Usage
```r
geom.regs(y, x, tol = 1e-07, type = 1, logged = FALSE, parallel = FALSE, maxiters = 100)
```

Arguments
- **y**: The dependent variable, count data.
- **x**: A matrix with the independent variables.
- **tol**: The tolerance value to terminate the Newton-Raphson algorithm.
- **type**: Type 1 refers to the case where the minimum is zero and type 2 for the case of the minimum being 1.
- **logged**: A boolean variable; it will return the logarithm of the pvalue if set to TRUE.
- **parallel**: Do you want this to be executed in parallel or not. The parallel takes place in C++, and the number of threads is defined by each system’s available cores.
- **maxiters**: The max number of iterations that can take place in each regression.
Many simple linear mixed model regressions

Details

Many simple geometric regressions are fitted.

Value

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

Author(s)

Stefanos Fafalios
R implementation and documentation: Stefanos Fafalios <stefanosfafalios@gmail.com>

See Also

poisson_only, prop.regs, score.geomregs

Examples

y <- rgeom(100, 0.6)
x <- matrix(rnorm(100 * 50), ncol = 50)
a <- geom.regs(y, x)
x <- NULL

rint.regs(y, x, id, tol = 1e-08, logged = FALSE, parallel = FALSE, maxiters = 100)

Arguments

y A numerical vector with the data. The subject values, the clustered data.
x A numerical matrix with data ,the independent variables.
id A numerical variable with 1, 2, ... indicating the subject. Unbalanced design is of course welcome.
tol The tolerance value to terminate the Newton-Raphson algorithm. This is set to $10^{-9}$ by default.
logged Should the p-values be returned (FALSE) or their logarithm (TRUE)?
parallel Do you want this to be executed in parallel or not. The parallel takes place in C++, and the number of threads is defined by each system’s available cores.
maxiters The max number of iterations that can take place in each regression.
Many simple linear regressions coefficients

Details

Many linear mixed models with a single covariate are fitted. We use Newton-Raphson as described in Demidenko (2013). The test statistic is the usual F-test. This model allows for random intercepts only.

Value

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

Author(s)

Stefanos Fafalios

R implementation and documentation: Stefanos Fafalios <stefanosfafalios@gmail.com>

References


See Also

rint.reg, allbetas.univglms, score.glms, logistic_only

Examples

```r
# not a so good example
y <- rnorm(100)
id <- sample(1:10, 100, replace = TRUE)
x <- matrix( rnorm(100 * 100), ncol = 100)
a <- rint.regs(y, x, id)
x <- NULL
```

Description

Simple linear regressions coefficients.

Usage

allbetas(y, x, pvalue = FALSE, logged = FALSE)
Many simple multinomial regressions

Arguments

y  A numerical vector with the response variable.

x  A matrix with the data, where rows denotes the observations and the columns contain the independent variables.

pvalue  If you want a hypothesis test that each slope (beta coefficient) is equal to zero set this equal to TRUE. It will also produce all the correlations between y and x.

logged  A boolean variable; it will return the logarithm of the pvalue if set to TRUE.

Value

A matrix with the constant (alpha) and the slope (beta) for each simple linear regression. If the p-value is set to TRUE, the correlation of each y with the x is calculated along with the relevant test statistic and its associated p-value.

Author(s)

Michail Tsagris

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

See Also

mvbetas, correls, univglms, colsums, colVars

Examples

x <- matrix( rnorm(100 * 50), ncol = 50 )
y <- rnorm(100)
r <- cor(y, x)  ## correlation of y with each of the xs
a <- allbetas(y, x)  ## the coefficients of each simple linear regression of y with x
x <- NULL

Many simple multinomial regressions

Description

Many simple multinomial regressions.

Usage

multinom.regs(y, x, tol = 1e-08, logged = FALSE, parallel = FALSE, maxiters = 100)
Many simple regressions for positive valued data

Arguments

- `y`: The dependent variable, either a numerical variable or a factor variable.
- `x`: A matrix with the independent variables.
- `tol`: The tolerance value to terminate the Newton-Raphson algorithm.
- `logged`: A boolean variable; it will return the logarithm of the p-value if set to TRUE.
- `parallel`: Do you want this to be executed in parallel or not. The parallel takes place in C++, and the number of threads is defined by each system’s available cores.
- `maxiters`: The maximum number of iterations that can take place in each regression.

Details

Many simple multinomial regressions are fitted.

Value

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

Author(s)

Stefanos Fafalios

R implementation and documentation: Stefanos Fafalios <stefanosfafalios@gmail.com>

See Also

- `poisson_only`
- `prop.regs`
- `score.geomregs`

Examples

```r
y <- rbinom(100, 2, 0.5)
x <- matrnorm(100, 100)
a <- multinom.regs(y, x)
x <- NULL
```

Description

Many simple regressions for positive valued data.

Usage

```r
normlog.regs(y, x, tol = 1e-08, logged = FALSE, parallel = FALSE, maxiters = 100)
gammaregs(y, x, tol = 1e-07, logged = FALSE, maxiters = 100)
invgauss.regs(y, x, tol = 1e-08, logged = FALSE, maxiters = 100)
```
Arguments

\(y\)  The dependent variable, a numerical variable with non negative numbers for the Gamma and inverse Gaussian regressions. For the Gaussian with a log-link zero values are allowed.

\(x\)  A matrix with the independent variables.

tol  The tolerance value to terminate the Newton-Raphson algorithm.

logged  A boolean variable; it will return the logarithm of the pvalue if set to TRUE.

parallel  Do you want this to be executed in parallel or not. The parallel takes place in C++, therefore you do not have the option to set the number of cores.

maxiters  The maximum number of iterations that can take place in each regression.

Details

Many simple Gamma, inverse Gaussian or Gaussian regressions with a log-link are fitted.

Value

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

Author(s)

Stefanos Fafalios and and Michail Tsagris

R implementation and documentation: Stefanos Fafalios <stefanosfafalios@gmail.com> and Michail Tsagris <mtsagris@uoc.gr>

References


See Also

\texttt{normlog.reg, score.glms, prop.regs, allbetas}

Examples

```r
## Not run:
y <- abs( rnorm(100) )
x <- matrnorm(100, 100)
a <- normlog.regs(y, x)
b <- glm(y ~ x[, 1], family = gaussian(log) )
anova(b, test= "F")
a[1, ]
a2 <- gammadregs(y, x)
a3 <- invgauss.regs(y, x)
```
Many tests for the dispersion parameter in Poisson distribution

```r
x <- NULL
## End(Not run)
```

**Description**

Many tests for the dispersion parameter in Poisson distribution.

**Usage**

```r
colpoisdisp.tests(y, alternative = "either", logged = FALSE)
colpois.tests(y, logged = FALSE)
```

**Arguments**

- `y`: A numerical matrix with count data, 0, 1,...
- `alternative`: Do you want to test specifically for either over or undersiprsion ("either"), overdispersion ("over") or undersispersion ("under")?
- `logged`: Set to TRUE if you want the logarithm of the p-value.

**Value**

A matrix with two columns, the test statistic and the (logged) p-value.

**Author(s)**

Michail Tsagris

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

**References**


Many two-way ANOVAs

See Also

poisson.mle, negbin.mle, poisson.anova, poisson.anovas, poisson_only

Examples

```r
y <- matrix(rnbinom(100* 50, 10, 0.6), ncol = 50)
a1 <- colpoisdisp.tests(y, "over")
b1 <- colpois.tests(y)

y <- matrix(rpois(100* 50, 10), ncol = 50)
a2 <- colpoisdisp.tests(y, "either")
b2 <- colpois.tests(y)
y <- NULL
```

Description

Many two-way ANOVAs.

Usage

twoway.anovas(y, x1, x2, interact = FALSE, logged = FALSE)

Arguments

- `y`: A matrix with the data, where the rows denote the observations (and the two groups) and the columns are the variables.
- `x1`: A numerical vector with 1s, 2s, 3s and so one indicating the two groups. Alternatively it can be a factor variable. This is the one factor.
- `x2`: A numerical vector with 1s, 2s, 3s and so one indicating the two groups. Alternatively it can be a factor variable. This is the other factor.
- `interact`: A boolean variable specifying whether you want to test for interaction.
- `logged`: Should the p-values be returned (FALSE) or their logarithm (TRUE)?

Details

The classical two-way ANOVA design is performed. Note that the design must be balanced. For every combination of values of the two factors, x1 and x2 the same number of observations must exist. If that’s not the case, regression models must be used.

Value

A matrix with the test statistic and the p-value of each test.
Many univariate generalised linear models

Author(s)

Michail Tsagris

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

References


See Also

ancovas, ftests, ttests

Examples

## Not run:
y <- as.matrix( rnorm(125) )
x1 <- rep(1:5, 25)
x2 <- rep(1:5, each = 25)
x1 <- factor(x1)
x2 <- factor(x2)
res<-anova( lm(y ~ x1 + x2) )
res<-twoway.anovas(y, x1, x2)
res<-anova( lm(y ~ x1*x2) )
res<-twoway.anovas(y, x1, x2, interact = TRUE)
y <- matrnorm(125, 100)
system.time( a1 <- twoway.anovas(y, x1, x2) )
system.time( a2 <- twoway.anovas(y, x1, x2, interact = TRUE) )
y <- NULL

## End(Not run)
Many univariate generalised linear models

Arguments

\( y \)
The dependent variable. It can be a factor or a numerical variable with two values only (binary logistic regression), a discrete valued vector (count data) corresponding to a poisson regression or a numerical vector with continuous values (normal regression).

\( x \)
A matrix with the data, where the rows denote the samples (and the two groups) and the columns are the variables. For the "univglms" only continuous variables are allowed. You are advised to standardise the data before hand to avoid numerical overflow or similar issues. If you see NaN in the outcome, this might be the case. For the "univglms2" categorical variables are allowed and hence this accepts data.frames. In this case, the categorical variables must be given as factor variables, otherwise you might get wrong results.

\( oiko \)
This can be either "normal", "poisson", "quasipoisson" or "binomial". If you are not sure leave it NULL and the function will check internally. However, you might have discrete data (e.g. years of age) and want to perform many simple linear regressions. In this case you should specify the family.

\( \text{logged} \)
A boolean variable; it will return the logarithm of the p-value if set to TRUE.

Details

If you specify no family of distributions the function internally checks the type of your data and decides on the type of regression to perform. The function is written in C++ and this is why it is very fast. It can accept thousands of predictor variables. It is useful for univariate screening. We provide no p-value correction (such as fdr or q-values); this is up to the user.

Value

A matrix with the test statistic and the p-value for each predictor variable.

Author(s)

Michail Tsagris

R implementation and documentation: Michail Tsagris <mtsagris@uoc.gr> and Manos Papadakis <papadam95@gmail.com>.

References


See Also

logistic_only,poisson_only,allbetas,correls,regression
Many univariate simple linear regressions

Examples

```r
## Not run:
x <- matrnorm(100, 50)
y <- rbinom(100, 1, 0.6)  # binary logistic regression
a1 <- univglm(y, x)
a2 <- glm(y ~ x[, 1], binomial)$deviance
a2 <- glm(y ~ 1, binomial)$null.dev - a2
x <- NULL

## End(Not run)
```

Description

It performs very many univariate simple linear regressions with or without categorical variables.

Usage

```r
regression(x, y, poia = NULL, logged = FALSE)
```

Arguments

- `x`: A data.frame or a matrix with the data, where the rows denote the samples (and the two groups) and the columns are the variables. A data frame is expected if you have categorical predictor variables. If you only have continuous predictor variables you should use the function `allbetas` instead as it is faster.
- `y`: The dependent variable; a numerical vector.
- `poia`: If the "x" is a data.frame and you know the indices of the columns which are categorical variables supply it here.
- `logged`: Do you want the logarithm of the p-values be returned? The default value is FALSE.

Details

Some parts of the function will be transferred in C++. It can accept thousands of predictor variables. It is useful for univariate screening. We provide no p-value correction (such as fdr or q-values); this is up to the user.

Value

A matrix with two columns, the test statistic value and its corresponding (logged) p-value.
Many univariate simple logistic and Poisson regressions

Author(s)

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

References


See Also

univglms, allbetas, correls, univglms, mvbetas

Examples

```r
y <- rnorm(150)
a <- regression(iris, y)
a
summary(lm(y ~ iris[, 5]) )  ## check the F-test
```

Description

It performs very many univariate simple binary logistic regressions.

Usage

```r
logistic_only(x, y, tol = 1e-09, b_values = FALSE)
poisson_only(x, y, tol = 1e-09, b_values = FALSE)
```

Arguments

- **x**
  A matrix with the data, where the rows denote the samples (and the two groups) and the columns are the variables. Currently only continuous variables are allowed.

- **y**
  The dependent variable; a numerical vector with two values (0 and 1) for the logistic regressions and a vector with many discrete values (count data) for the Poisson regressions.

- **tol**
  The tolerance value to terminate the Newton-Raphson algorithm.

- **b_values**
  Do you want the values of the coefficients returned? If yes, set this to TRUE.
Many univariate simple logistic and Poisson regressions

Details

The function is written in C++ and this is why it is very fast. It can accept thousands of predictor variables. It is useful for univariate screening. We provide no p-value correction (such as fdr or q-values); this is up to the user.

Value

A vector with the deviance of each simple binary logistic regression model for each predictor variable.

Author(s)

Manos Papadakis <papadakm95@gmail.com>

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

References


See Also

univglms,score.glms,prop.regs,quasi.poisson_only,allbetas,correls,regression

Examples

```r
## Not run:
## 300 variables, hence 300 univariate regressions are to be fitted
x <- matrix( rnorm(100 * 300), ncol = 300 )

## 100 observations in total
y <- rbinom(100, 1, 0.6)  ## binary logistic regression
a1 <- logistic_only(x, y)

a2 <- glm(y ~ x[, 1], binomial)$deviance
a2 <- as.vector(a2)

y <- rpois(100, 10)
a1 <- poisson_only(x, y)

a1 <- x <- NULL

## End(Not run)
```
**Many univariate simple quasi poisson regressions**

Many univariate simple poisson regressions

**Description**

It performs very many univariate simple poisson regressions.

**Usage**

```r
quasi.poisson_only(x, y, tol = 1e-09, maxiters = 100)
```

**Arguments**

- `x` A matrix with the data, where the rows denote the samples (and the two groups) and the columns are the variables. Currently only continuous variables are allowed.
- `y` The dependent variable; a numerical vector with many discrete values (count data).
- `maxiters` The maximum number of iterations after which the Newton-Raphson algorithm is terminated.
- `tol` The tolerance value to terminate the Newton-Raphson algorithm.

**Details**

The function is written in C++ and this is why it is very fast. It can accept thousands of predictor variables. It is useful for univariate screening. We provide no p-value correction (such as fdr or q-values); this is up to the user.

**Value**

A matrix with the deviance and the estimated phi parameter (dispersion parameter) of each simple poisson regression model for each predictor variable.

**Author(s)**

Manos Papadakis <papadakm95@gmail.com> and Stefanos Fafalios <stefanosfafalios@gmail.com>

R implementation and documentation: Michail Tsagris <mtsagris@uoc.gr>, Manos Papadakis <papadakm95@gmail.com> and Stefanos Fafalios <stefanosfafalios@gmail.com>.

**References**

Many Welch’s F-tests

See Also

poisson_only, univglms, logistic_only, allbetas, regression

Examples

## 200 variables, hence 200 univariate regressions are to be fitted
x <- matrix( rnorm(100 * 200), ncol = 200 )
y <- rpois(100, 10)
system.time( poisson_only(x, y) )
b1 <- poisson_only(x, y)
b2 <- quasi.poisson_only(x, y)

b1<-b2<-x<-y<-NULL

Many Welch’s F-tests

Description

Many Welch’s F-tests.

Usage

colanovas(y, x, logged = FALSE)

Arguments

y
A numerical vector with the dependent variable.
x
A matrix with the data, where the rows denote the samples (and the two groups) and the columns are the variables. This must be a matrix with the categorical variables as numbers, starting from 1. Welch’s F-test is performed for each variable.
logged
A boolean variable; it will return the logarithm of the p-value if set to TRUE.

Details

For each categorical variable in the x matrix Welch’s F-test is performed. This is the opposite of ftests, where there are many dependent variables and one categorical variable.

Value

A matrix with the test statistic and the p-value for each predictor variable.

Author(s)

Michail Tsagris

R implementation and documentation: Michail Tsagris <mtsagris@uoc.gr> and Manos Papadakis <papadakm95@gmail.com>.
References


See Also

regression, ftests, allbetas, correls

Examples

```r
y <- rnorm(100)
x <- matrix( rbinom(100 * 50, 2, 0.5) + 1 , ncol = 50)
a <- colanovas(y, x)
x <- NULL
```

Match

<table>
<thead>
<tr>
<th>Match</th>
<th>Match</th>
</tr>
</thead>
</table>

Description

Return the positions of its first argument that matches in its second.

Usage

`Match(x, key=NULL)`

Arguments

- `x` A numeric vector.
- `key` The value/vector for searching in vector `x`. For now let it NULL. `don't use it!`

Details

This function implements the R's `match` function. This version basically calculates the `match(x, sort(unique(x)))` for now. Do not use the argument `key`!

Value

Returns the position/positions of the given key/keys in the `x` vector.

Author(s)

Manos Papadakis
R implementation and documentation: Manos Papadakis <papadakm95@gmail.com>
**Matrix multiplication**

**See Also**

* match

**Examples**

```r
y <- rnorm(100)
a <- Match(y)
b <- 50
all.equal(as.vector(a), as.vector(b))
```

---

**Description**

Matrix multiplication, Cross and Tcross product.

**Usage**

```r
mat.mult(x, y)
Crossprod(x, y)
Tcrossprod(x, y)
```

**Arguments**

- `x`: A numerical matrix.
- `y`: A numerical matrix.

**Details**

The functions perform matrix multiplication, cross product and transpose cross product. There are faster(!) than R’s function for large matrices. Depending on the computer, maybe higher dimensions are required for the function to make a difference. The function runs in parallel in C++.

**Value**

A matrix, the result of the matrix multiplication.

**Author(s)**

Manos Papadakis

R implementation and documentation: Manos Papadakis <papadakm95@gmail.com>

**See Also**

* transpose, colsums
Matrix with all pairs of t-tests

Examples

```r
## Not run:
x <- matrnorm(100, 100)
y <- matrnorm(100, 100)
a <- x
b <- mat.mult(x, y)
b <- Crossprod(x, y)
b <- Tcrossprod(x, y)
x <- NULL
y <- NULL
b <- NULL

## End(Not run)
```

Description

Matrix with all pairs of t-tests.

Usage

```r
allttests(x, y = NULL, ina, logged = FALSE)
ttests.pairs(x, logged = FALSE)
```

Arguments

- **x**: A numerical matrix with the data.
- **y**: For the case of "all.tests", if you have the second group or sample provide it here, otherwise leave it NULL. For the case of "ttests.pairs" this is not required.
- **ina**: If you have the data in one matric then provide this indicator variable separating the samples. This numerical vector must contain 1s and 2s only as values. For the case of "ttests.pairs" this is not required.
- **logged**: Should the p-values be returned (FALSE) or their logarithm (TRUE)?

Details

The function does all the pairwise t-tests assuming unequal variances (Welch's t-test). The "all.ttests" does all the pairs formed by "cutting" the matrices x and y in two and everything between them. The "ttests.pairs" accepts a matrix x and does all the pairs of t-tests. This is similar to the correlation matrix style.
Matrix with G-square tests of independence

**Value**

A list including:

- **stat**: A matrix with t-test statistic for each pair of variables.
- **pvalue**: A matrix with the corresponding p-values.
- **dof**: A matrix with the relevant degrees of freedom.

**Author(s)**

Michail Tsagris

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

**See Also**

`ttests, fttests, ttest, g2Test_univariate`

**Examples**

```r
x <- as.matrix( iris[1:100, 1:4] )
ina <- as.numeric(iris[1:100, 5])
a <- allttests(x, ina = ina)
b <- ttests.pairs(x)  ## less tests
```

---

Matrix with G-square tests of independence

Matrix with G-square tests of independence

**Description**

Matrix with G-square tests of independence with and without permutations.

**Usage**

```r
g2Test_univariate(data, dc)
g2Test_univariate_perm(data, dc, nperm)
chi2Test_univariate(data, dc)
```

**Arguments**

- **data**: A numerical matrix with the data. **The minimum must be 0, otherwise the function can crash or will produce wrong results.** The data must be consecutive numbers.
- **dc**: A numerical value equal to the number of variables (or columns of the data matrix) indicating the number of distinct, unique values (or levels) of each variable. Make sure you give the correct numbers here, otherwise the degrees of freedom will be wrong.
The number of permutations. The permutations test is slower than without permutations and should be used with small sample sizes or when the contingency tables have zeros. When there are few variables, R’s "chisq.test" function is faster, but as the number of variables increase the time difference with R’s procedure becomes larger and larger.

Details

The function does all the pairwise $G^2$ test of independence and gives the position inside the matrix. The user must build the associations matrix now, similarly to the correlation matrix. See the examples of how to do that. The p-value is not returned, we live this to the user. See the examples of how to obtain it.

Value

A list including:

- `statistic`: The $G^2$ or chi$^2$ test statistic for each pair of variables.
- `pvalue`: This is returned when you have selected the permutation based $G^2$ test.
- `x`: The row or variable of the data.
- `y`: The column or variable of the data.
- `df`: The degrees of freedom of each test.

Author(s)

Giorgos Borboudakis. The permutation version used a C++ code by John Burkardt.

R implementation and documentation: Manos Papadakis <papadakm95@gmail.com>.

References


See Also

`g2Test, g2Test_perm, correls, univglms`

Examples

```r
nvalues <- 3
nvars <- 10
nsamples <- 2000
data <- matrix(sample(0:(nvalues - 1), nvars * nsamples, replace = TRUE), nsamples, nvars)
dc <- rep(nvalues, nvars)
system.time( g2Test_univariate(data = data, dc = dc) )
a <- g2Test_univariate(data = data, dc = dc)
```
Mean - Median absolute deviation of a vector

Description

Mean - Median absolute deviation of a vector.

Usage

mad2(x, method = "median", na.rm = FALSE)
Mad(x, method = "median", na.rm = FALSE)

Arguments

\(x\) A numerical vector.

method A character vector with values "median", for median absolute deviation or "mean", for mean absolute deviation.

na.rm A logical value TRUE/FALSE to remove NAs.

Value

The mean absolute deviation.

Author(s)

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

See Also

colMads, Median, colMedians

Examples

\(x \leftarrow \text{Rnorm}(1000)\)

Mad(x)
mad(x)
Median of a vector

Description

Median of a vector.

Usage

med(x, na.rm=FALSE)
Median(x, na.rm=FALSE)

Arguments

x
A numerical vector.

na.rm
TRUE or FALSE for remove NAs if exists.

Details

The function is written in C++ and this is why it is very fast.

Value

The median of the vector of a numbers.

Author(s)

Manos Papadakis <papadakm95@gmail.com>

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

See Also

nth, colMedians

Examples

x <- rnorm(1000)
a1 <- Median(x)
a2 <- median(x)
Minima and maxima of two vectors/matrices

Minima and maxima of two vectors/matrices

Description

Minima and maxima of two vectors/matrices.

Usage

Pmax(x, y, na.rm = FALSE)
Pmin(x, y, na.rm = FALSE)
Pmin_Pmax(x, y, na.rm = FALSE)

Arguments

x
A numerical vector with numbers.
y
A numerical vector with numbers.
na.rm
TRUE or FALSE for removing NAs if exists.

Details

The parallel minima or maxima are returned. This are the same as the base functions pmax and pmin.

Value

A numerical vector/matrix with numbers, whose length is equal to the length of the initial vectors/matrices containing the maximum or minimum between each pair.

Author(s)

Manos Papadakis
R implementation and documentation: Manos Papadakis <papadakm95@gmail.com>.

See Also

colSort, rowSort, Sort, colMins

Examples

x <- rnorm(10)
y <- rnorm(10)
res <- Pmax(x, y)
a <- pmax(x, y)
res <- Pmin(x, y)
b <- pmin(x, y)
res <- Pmin_Pmax(x, y) == c(a, b)
a <- b <- x <- y <- NULL
Minimum and maximum of a vector

Description

Minimum and maximum of a vector.

Usage

\[
\text{min_max}(x, \text{index}=\text{FALSE}, \text{percent} = \text{FALSE})
\]

Arguments

- **x**: A numerical vector with data. NAs are handled naturally.
- **index**: A boolean value for the indices of the minimum and the maximum value.
- **percent**: A boolean value for the percent of the positive and negative numbers.

Value

A vector with the relevant values, min and max.

Author(s)

Manos Papadakis

R implementation and documentation: Manos Papadakis <papadakm95@gmail.com>.

See Also

- `rowMins`, `rowMaxs`, `nth`, `colrange`, `colMedians`, `colSort`, `rowSort`

Examples

\[
x <- \text{rnorm}(100 * 500)
s1 <- \text{min_max}(x)
s2 <- c(\text{min}(x), \text{max}(x))
\]
Minimum and maximum frequencies of a vector

Description
Minimum and maximum frequencies of a vector.

Usage
freq.min(x, na.rm = FALSE)
freq.max(x, na.rm = FALSE)

Arguments
x
A numerical/integer vector with data but without NAs.
na.rm
TRUE or FALSE for remove NAs if exists.

Details
Those functions are the same with max(table(x)) or min(table(x)) but with one exception. freq.min and freq.max will return also which value has the minimum/maximum frequency. More Efficient than max(table(x)) or min(table(x)).

Value
A vector with 2 values, the value with minimum/maximum frequency and the frequency.

Author(s)
Manos Papadakis
R implementation and documentation: Manos Papadakis <papadakm95@gmail.com> and Marios Dimitriadis <kmdimitriadis@gmail.com>.

See Also
rowMins, rowMaxs, nth, colrange, colMedians, colSort, rowSort

Examples
x <- rnorm(100)
f1 <- freq.min(x)
f2 <- freq.max(x)
# f1r <- min(table(x))
# f2r <- max(table(x))
# f1[2]==f1r  ## the frequencies are the same
# f2[2]==f2r  ## the frequencies are the same
MLE for multivariate discrete data

Description

MLE for multivariate discrete data.

Usage

```r
multinom.mle(x)
dirimultinom.mle(x, tol = 1e-07)
colpoisson.mle(x)
colgeom.mle(x, type = 1)
```

Arguments

- **x**: A matrix with discrete valued non-negative data.
- **tol**: the tolerance level to terminate the Newton-Raphson algorithm for the Dirichlet multinomial distribution.
- **type**: This is for the geometric distribution only. Type 1 refers to the case where the minimum is zero and type 2 for the case of the minimum being 1.

Details

For the Poisson and geometric distributions we simply fit independent Poisson and geometric distributions respectively.

Value

A list including:

- **loglik**: A vector with the value of the maximised log-likelihood.
- **param**: A vector of the parameters.

Author(s)

Michail Tsagris

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

References


MLE of (hyper-)spherical distributions

See Also

poisson.mle, zip.mle, ztp.mle, negbin.mle, poisson.nb

Examples

```r
x <- t(rmultinom(1000, 20, c(0.4, 0.5, 0.1)))
res<-multinom.mle(x)
res<-colpoisson.mle(x)
x <- NULL
```

Description

MLE of (hyper-)spherical distributions.

Usage

```r
vmf.mle(x, tol = 1e-07)
multivmf.mle(x, ina, tol = 1e-07, ell = FALSE)
acg.mle(x, tol = 1e-07)
iag.mle(x, tol = 1e-07)
```

Arguments

- `x`: A matrix with directional data, i.e. unit vectors.
- `ina`: A numerical vector with discrete numbers starting from 1, i.e. 1, 2, 3, 4,... or a factor variable. Each number denotes a sample or group. If you supply a continuous valued vector the function will obviously provide wrong results.
- `ell`: This is for the multivmf.mle only. Do you want the log-likelihood returned? The default value is TRUE.
- `tol`: The tolerance value at which to terminate the iterations.

Details

For the von Mises-Fisher, the normalised mean is the mean direction. For the concentration parameter, a Newton-Raphson is implemented. For the angular central Gaussian distribution there is a constraint on the estimated covariance matrix; its trace is equal to the number of variables. An iterative algorithm takes place and convergence is guaranteed. Newton-Raphson for the projected normal distribution, on the sphere, is implemented as well. Finally, the von Mises-Fisher distribution for groups of data is also implemented.
Value

For the von Mises-Fisher a list including:

- `loglik`: The maximum log-likelihood value.
- `mu`: The mean direction.
- `kappa`: The concentration parameter.

For the multi von Mises-Fisher a list including:

- `loglik`: A vector with the maximum log-likelihood values if `ell` is set to TRUE. Otherwise NULL is returned.
- `mi`: A matrix with the group mean directions.
- `ki`: A vector with the group concentration parameters.

For the angular central Gaussian a list including:

- `iter`: The number if iterations required by the algorithm to converge to the solution.
- `cova`: The estimated covariance matrix.

For the spherical projected normal a list including:

- `iters`: The number of iteration required by the Newton-Raphson.
- `mesi`: A matrix with two rows. The first row is the mean direction and the second is the mean vector. The first comes from the second by normalising to have unit length.
- `param`: A vector with the elements, the norm of mean vector, the log-likelihood and the log-likelihood of the spherical uniform distribution. The third value helps in case you want to do a log-likelihood ratio test for uniformity.

Author(s)

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

References


See Also

racg,vm.mle,rvmf
Examples

```r
m <- c(0, 0, 0, 0)
s <- cov(iris[, 1:4])
x <- racg(100, s)
mod <- acg.mle(x)
res <- cov2cor(mod$cova)  ## estimated covariance matrix turned into a correlation matrix
res <- cov2cor(s)  ## true covariance matrix turned into a correlation matrix
x <- rbind( rvmf(100, rnorm(4), 10), rvmf(100, rnorm(4), 20) )
a <- multivmf.mle(x, rep(1:2, each = 100) )
```

Description

MLE of continuous univariate distributions defined on the positive line.

Usage

```
gammamle(x, tol = 1e-09)
chisq.mle(x, tol = 1e-09)
weibull.mle(x, tol = 1e-09, maxiters = 100)
lomax.mle(x, tol = 1e-09)
foldnorm.mle(x, tol = 1e-09)
betaprime.mle(x, tol = 1e-09)
logcauchy.mle(x, tol = 1e-09)
loglogistic.mle(x, tol = 1e-09)
halfnorm.mle(x)
invgauss.mle(x)
lognorm.mle(x)
pareto.mle(x)
expmle(x)
exp2.mle(x)
maxboltz.mle(x)
rayleigh.mle(x)
normlog.mle(x)
lindley.mle(x)
```

Arguments

- `x` A vector with positive valued data (zeros are not allowed).
- `tol` The tolerance level up to which the maximisation stops; set to 1e-09 by default.
- `maxiters` The maximum number of iterations the Newton-Raphson will perform.
MLE of continuous univariate distributions defined on the positive line

Details

Instead of maximising the log-likelihood via a numerical optimiser we have used a Newton-Raphson algorithm which is faster. See wikipedia for the equations to be solved. For the t distribution we need the degrees of freedom and estimate the location and scatter parameters. If you want to to fit an inverse gamma distribution simply do "gamma.mle(1/x)". The log-likelihood and the parameters are for the inverse gamma.

The "normlog.mle" is simply the normal distribution where all values are positive. Note, this is not log-normal. It is the normal with a log link. Similarly to the inverse gaussian distribution where the mean is an exponentiated. This comes from the GLM theory.

Value

Usually a list with three elements, but this is not for all cases.

- `iters` The number of iterations required for the Newton-Raphson to converge.
- `loglik` The value of the maximised log-likelihood.
- `param` The vector of the parameters.

Author(s)

Michail Tsagris

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

References


You can also check the relevant wikipedia pages for these distributions.

See Also

zip.mle, normal.mle, beta.mle
Examples

```r
x <- rgamma(100, 3, 4)
system.time( for (i in 1:20) gammamle(x) )
## system.time( for (i in 1:20) fitdistr(x,"gamma") )
a <- glm(x ~ 1, gaussian(log) )
res<-normlog.mle(x)
```

Description

MLE of continuous univariate distributions defined on the real line.

Usage

```r
normal.mle(x)
gumbel.mle(x, tol = 1e-09)
cauchy.mle(x, tol = 1e-09)
logistic.mle(x, tol = 1e-07)
ct.mle(x, tol = 1e-09)
tmle(x, v = 5, tol = 1e-08)
wigner.mle(x, tol = 1e-09)
laplace.mle(x)
```

Arguments

- `x` A numerical vector with data.
- `v` The degrees of freedom of the t distribution.
- `tol` The tolerance level up to which the maximisation stops set to 1e-09 by default.

Details

Instead of maximising the log-likelihood via a numerical optimiser we have used a Newton-Raphson algorithm which is faster. See wikipedia for the equation to be solved. For the t distribution we need the degrees of freedom and estimate the location and scatter parameters.

The Cauchy is the t distribution with 1 degree of freedom. If you want to fit such a distribution used the cauchy.mle and not the tmle with 1 degree of freedom as it’s faster. The Laplace distribution is also called double exponential distribution.

The wigner.mle refers to the wigner semicircle distribution.
MLE of count data (univariate discrete distributions)

Value

Usually a list with three elements, but this is not for all cases.

- **iters**: The number of iterations required for the Newton-Raphson to converge.
- **loglik**: The value of the maximised log-likelihood.
- **param**: The vector of the parameters.

Author(s)

Michail Tsagris

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

References


See Also

zip.mle, gammamle, vm.mle

Examples

```r
x <- rt(1000,10)
a <- ct.mle(x)
res<-tmle(x, v = a$nu)
res<-cauchy.mle(x)
res<-normal.mle(x)
res<-logistic.mle(x)
res<-gumbel.mle(x)
```

Description

MLE of count data.
MLE of count data (univariate discrete distributions)

Usage

zip.mle(x, tol = 1e-09)
ztp.mle(x, tol = 1e-09)
negbin.mle(x, type = 1, tol = 1e-09)
binom.mle(x, N = NULL, tol = 1e-07)
borel.mle(x)
geom.mle(x, type = 1)
logseries.mle(x, tol = 1e-09)
poisson.mle(x)
betageom.mle(x, tol = 1e-07)
betabinom.mle(x, N, tol = 1e-07)

Arguments

x
A vector with discrete valued data.

type
This argument is for the negative binomial and the geometric distribution. In the negative binomial you can choose which way you prefer. Type 1 is for small sample sizes, whereas type 2 is for larger ones as is faster. For the geometric it is related to its two forms. Type 1 refers to the case where the minimum is zero and type 2 for the case of the minimum being 1.

N
This is for the binomial distribution only, specifying the total number of successes. If NULL, it is estimated by the data. It can also be a vector of successes.

tol
The tolerance level up to which the maximisation stops set to 1e-09 by default.

Details

Instead of maximising the log-likelihood via a numerical optimiser we used a Newton-Raphson algorithm which is faster.

See wikipedia for the equation to be solved in the case of the zero inflated distribution. https://en.wikipedia.org/wiki/Zero-inflated_model. In order to avoid negative values we have used link functions, log for the $\lambda$ and logit for the $\pi$ as suggested by Lambert (1992). As for the zero truncated Poisson see https://en.wikipedia.org/wiki/Zero-truncated_Poisson_distribution.

zip.mle is for the zero inflated Poisson, whereas ztp.mle is for the zero truncated Poisson distribution.

Value

The following list is not inclusive of all cases. Different functions have different names. In general a list including:

mess
This is for the negbin.mle only. If there is no reason to use the negative binomial distribution a message will appear, otherwise this is NULL.

iters
The number of iterations required for the Newton-Raphson to converge.

loglik
The value of the maximised log-likelihood.

prob
The probability parameter of the distribution. In some distributions this argument might have a different name. For example, param in the zero inflated Poisson.
MLE of distributions defined in the (0, 1) interval

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

References

See Also
poisson_only, colrange

Examples
x <- rpois(100, 2)
res<-zip.mle(x)
res<-poisson.mle(x)
## small difference in the two log-likelihoods as expected.
x <- rpois(100, 10)
x[x == 0 ] <- 1
res<-ztp.mle(x)
res<-poisson.mle(x)
## significant difference in the two log-likelihoods.
x <- rnbinom(100, 10, 0.6)
res<-poisson.mle(x)
res<-negbin.mle(x)

MLE of distributions defined in the (0, 1) interval

Description
MLE of distributions defined in the (0, 1) interval.

Usage
beta.mle(x, tol = 1e-09)
ibeta.mle(x, tol = 1e-09)
logitnorm.mle(x)
hsecant01.mle(x, tol = 1e-09)
MLE of distributions defined in the (0, 1) interval

Arguments

x A numerical vector with proportions, i.e. numbers in (0, 1) (zeros and ones are not allowed).
tol The tolerance level up to which the maximisation stops.

Details

Maximum likelihood estimation of the parameters of the beta distribution is performed via Newton-Raphson. The distributions and hence the functions does not accept zeros. "logitnorm.mle" fits the logistic normal, hence no newton-Raphson is required and the "hypersecant01.mle" uses the golden ratio search as is it faster than the Newton-Raphson (less calculations)

Value

A list including:

iters The number of iterations required by the Newton-Raphson.
loglik The value of the log-likelihood.
param The estimated parameters. In the case of "hypersecant01.mle" this is called "theta" as there is only one parameter.

Author(s)

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

See Also

diri.nr2,

Examples

x <- rbeta(1000, 1, 4)
system.time( for(i in 1:1000) beta.mle(x) )
res<-beta.mle(x)
res<-ibeta.mle(x)

x <- runif(1000)
res<-hsecant01.mle(x)
res<-logitnorm.mle(x)
res<-ibeta.mle(x)

x <- rbeta(1000, 2, 5)
x[sample(1:1000, 50)] <- 0
res<-ibeta.mle(x)
MLE of some circular distributions

Description

MLE of some circular distributions.

Usage

\begin{verbatim}
vm.mle(x, tol = 1e-09)
spml.mle(x, tol = 1e-09, maxiters = 100)
wrapcauchy.mle(x, tol = 1e-09)
\end{verbatim}

Arguments

- **x**: A numerical vector with the circular data. They must be expressed in radians. For the "spml.mle" this can also be a matrix with two columns, the cosinus and the sinus of the circular data.
- **tol**: The tolerance level to stop the iterative process of finding the MLEs.
- **maxiters**: The maximum number of iterations to implement.

Details

The parameters of the von Mises, the bivariate angular Gaussian and wrapped Cauchy distributions are estimated. For the Wrapped Cauchy, the iterative procedure described by Kent and Tyler (1988) is used. As for the von Mises distribution, we use a Newton-Raphson to estimate the concentration parameter. The angular Gaussian is described, in the regression setting in Presnell et al. (1998).

Value

A list including:

- **iters**: The iterations required until convergence. This is returned in the wrapped Cauchy distribution only.
- **loglik**: The value of the maximised log-likelihood.
- **param**: A vector consisting of the estimates of the two parameters, the mean direction for both distributions and the concentration parameter kappa and the rho for the von Mises and wrapped Cauchy respectively.
- **gamma**: The norm of the mean vector of the angular Gaussian distribution.
- **mu**: The mean vector of the angular Gaussian distribution.

Author(s)

Michail Tsagris and Stefanos Fafalios

R implementation and documentation: Michail Tsagris <mtsagris@uoc.gr> and Stefanos Fafalios <stefanosfafalios@gmail.com>
MLE of the inverted Dirichlet distribution

References

See Also
vmf.mle,rvonmises,rvmf

Examples

```r
y <- rcauchy(100, 3, 1)
x <- y
res<-vm.mle(x)
res<-spml.mle(x)
res<-wrapcauchy.mle(x)
x <- NULL
```

MLE of the inverted Dirichlet distribution

MLE of the inverted Dirichlet distribution

Description
MLE of the inverted Dirichlet distribution.

Usage

```r
invdir.mle(x, tol = 1e-09)
```

Arguments

- `x` A matrix with strictly positive data (no zeros are allowed).
- `tol` The tolerance level up to which the maximisation stops.

Details
Maximum likelihood estimation of the parameters of the inverted is performed via Newton-Raphson. We took the initial values suggested by Bdiri T. and Bouguila N. (2012) and modified them a bit.
MLE of the multivariate (log-) normal distribution

Value
A list including:

- **iters**  The number of iterations required by the Newton Raphson.
- **loglik**  The value of the log-likelihood.
- **param**  The estimated parameters.

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

References

See Also
diri.nr2, multinom.mle

Examples
```r
x <- as.matrix(iris[, 1:4])
system.time( for(i in 1:100) invdir.mle(x) )
res<-invdir.mle(x)
```

MLE of the multivariate (log-) normal distribution

MLE of the multivariate (log-) normal distribution

Description
MLE of the multivariate (log-) normal distribution.

Usage
```r
mvnorm.mle(x)
mvlnorm.mle(x)
```

Arguments
- **x**  A matrix with numerical data.
Details

The mean vector, covariance matrix and the value of the log-likelihood of the multivariate normal or log-normal distribution is calculated. For the log-normal distribution we also provide the expected value and the covariance matrix.

Value

A list including:

- **loglik**: The log-likelihood multivariate distribution.
- **mu**: The mean vector.
- **sigma**: The covariance matrix.
- **m**: The expected mean vector of the multivariate log-normal distribution.
- **s**: The expected covariance matrix of the multivariate log-normal distribution.

Author(s)

Michail Tsagris

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

References


http://isi.cbs.nl/iamamember/CD2/pdf/329.PDF


See Also

multinom.mle, dmvnorm, gaussian.nb

Examples

```r
x <- matrnorm(100, 4)
res <- mvnorm.mle(x)
x <- NULL
```
ML of the multivariate t distribution

Description

ML of the multivariate t distribution.

Usage

mvt.mle(x, v = 5, tol = 1e-07)

Arguments

x
A matrix with numerical data.
v
The degrees of freedom. Must be a positive number, greater than zero.
tol
The tolerance value to terminate the EM algorithm.

Details

The location vector, scatter matrix and the value of the log-likelihood is calculated.

Value

A list including:

iters
The number of iterations required for the EM algorithm to converge.
loglik
The value of the maximised log-likelihood.
location
The location vector.
scatter
The scatter matrix.

Author(s)

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

References


See Also

mvnorm.mle, dmvnorm, gaussian.nb
MLE of the ordinal model without covariates

**Examples**

\[
\begin{align*}
  x & \leftarrow \text{matrnorm}(100, 4) \\
  \text{res} & \leftarrow \text{mvnorm.mle}(x) \\
  \text{res} & \leftarrow \text{mvt.mle}(x, v = 5) \\
  \text{res} & \leftarrow \text{mvt.mle}(x, v = 100)
\end{align*}
\]

**Description**

MLE of the ordinal model without covariates.

**Usage**

```
ordinal.mle(y, link = "logit")
```

**Arguments**

- `y` A numerical vector with values 1, 2, 3,..., not zeros, or an ordered factor.
- `link` This can either be "logit" or "probit". It is the link function to be used.

**Details**

Maximum likelihood of the ordinal model (proportional odds) is implemented. See for example the "polr" command in R or the examples.

**Value**

A list including:

- `loglik` The log-likelihood of the model.
- `a` The intercepts (threshold coefficients) of the model.

**Author(s)**

Manos Papadakis

R implementation and documentation: Manos Papadakis <papadakm95@gmail.com>.

**References**


**See Also**

`beta.mle`, `diri.nr2`
**Examples**

```r
y <- factor( rbinom(100,3,0.5), ordered = TRUE )
res<-ordinal.mle(y)
res<-ordinal.mle(y, link = "probit")
```

**Description**

MLE of the tobit model.

**Usage**

```r
tobit.mle(y, tol = 1e-09)
```

**Arguments**

- `y`: A vector with positive valued data and zero values. If there are no zero values, a simple normal model is fitted in the end.
- `tol`: The tolerance level up to which the maximisation stops; set to 1e-09 by default.

**Details**

The tobit model is useful for (univariate) positive data with left censoring at zero. There is the assumption of a latent variable. The values of that variable which are positive coincide with the observed values. If some values are negative, they are left censored and the observed values are zero. Instead of maximising the log-likelihood via a numerical optimiser we have used a Newton-Raphson algorithm which is faster.

**Value**

A list with three elements including

- `iters`: The number of iterations required for the Newton-Raphson to converge.
- `loglik`: The value of the maximised log-likelihood.
- `param`: The vector of the parameters.

**Author(s)**

Michail Tsagris

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

---

**MLE of the tobit model**

**MLE of the tobit model**
References

https://en.wikipedia.org/wiki/Tobit_model

See Also

gammamle, normal.mle

Examples

x <- rnorm(300, 3, 5)
x[ x < 0 ] <- 0  ## left censoring. Values below zero become zero
system.time( for (i in 1:100) tobit.mle(x) )

Description

Moment and maximum likelihood estimation of variance components.

Usage

rint.mle(x, ina, ranef = FALSE, tol = 1e-09, maxiters = 100)
varcomps.mom(x, ina)
varcomps.mle(x, ina, tol = 1e-09)

Arguments

x A numerical vector with the data.
ranef Should the random effects be returned as well? The default value is FALSE.
ina A numerical vector with 1s, 2s, 3s and so one indicating the two groups. Be careful, the function is designed to accept numbers greater than zero. Alternatively it can be a factor variable.
tol The tolerance level to terminate the golden ratio search. the default value is 10^(-9).
maxiters The maximum number of iterations Newton-Raphson will implement.
Details

Note that the "varcomps.mle" and "varcomp.mom" work for balanced designs only, i.e. for each subject the same number of measurements have been taken. The "rint.mle" works for both the balanced and unbalanced designs.

The variance components, the variance of the between measurements and the variance of the within are estimated using moment estimators. The "colvarcomsp.mom" is the moment analogue of a random effects model which uses likelihood estimation ("colvarcomps.mle"). It is much faster, but can give negative variance of the random effects, in which case it becomes zero.

The maximum likelihood version is a bit slower (try yourselves to see the difference), but statistically speaking is to be preferred when small samples are available. The reason why it is only a little bit slower and not a lot slower as one would imagine is because we are using a closed formula to calculate the two variance components (Demidenko, 2013, pg. 67-69). Yes, there are closed formulas for linear mixed models.

Value

For the "varcomps.mom": A vector with 5 elements, The MSE, the estimate of the between variance, the variance components ratio and a 95% confidence for the ratio.

For the "varcomps.mle": a list with a single component called "info". That is a matrix with 3 columns, The MSE, the estimate of the between variance and the log-likelihood value. If ranef = TRUE a list including "info" and an extra component called "ranef" containing the random effects. It is a matrix with the same number of columns as the data. Each column contains the random effects of each variable.

Author(s)

Michail Tsagris and Manos Papadakis

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

References


See Also

colvarcomps.mle, rint.reg, rint.regbx
Examples

## example from Montgomery, pages 514-517
```
x <- c(98, 97, 99, 96, 91, 90, 93, 92, 96, 95, 97, 95, 95, 96, 99, 98)
ina <- rep(1:4, each = 4)
res <- varcomps.mom(x, ina)
res <- varcomps.mle(x, ina)
```

Multi-sample tests for vectors

Description

Multi-sample tests for vectors.

Usage

```r
ftest(x, ina, logged = FALSE)
anova1(x, ina, logged = FALSE)
kruskal.test(x, ina, logged = FALSE)
var2test(x, y, alternative = "unequal", logged = FALSE)
mcnemar(x, y, logged = FALSE)
ttest2(x, y, paired = FALSE, logged = FALSE)
cqtest(x, treat, block, logged = FALSE)
block.anova(x, treat, block, logged = FALSE)
twoway.anova(y, x1, x2, interact = FALSE, logged = FALSE)
```

Arguments

- `x` A numerical vector with the data.
- `y` A numerical vector with the data.
- `ina` A numerical vector with 1s, 2s, 3s and so one indicating the two groups. Be careful, the function is designed to accept numbers greater than zero. Alternatively it can be a factor variable.
- `paired` This is for the two sample t-test only and is TRUE or FALSE specifying whether the two samples are paired or not.
- `alternative` This can either be "unequal", "greater" or "less".
- `treat` In the case of the blocking ANOVA and Cochran’s Q test, this argument plays the role of the "ina" argument.
- `block` This item (in the blocking ANOVA and Cochran’s Q test) denotes the subjects which are the same. Similarly to "ina" a numeric vector with 1s, 2s, 3s and so on.
- `x1` The first factor in the two way ANOVA.
- `x2` The second factor in the two way ANOVA. The order is not important.
- `interact` Should interaction in the two way ANOVA be included? The default value is FALSE (no interaction).
- `logged` Should the p-values be returned (FALSE) or their logarithm (TRUE)?
Multi-sample tests for vectors

Details

The Welch’s F-test (without assuming equal variances) is performed with the "ftest" function. The "anova" function perform the classical (Fisher’s) one-way analysis of variance (ANOVA) which assumes equal variance across the groups. The "kruskaltest" performs the Kruskal-Wallis non-parametric alternative to analysis of variance test. The "var2tests" implement the classical F test for the equality of two sample variances. The "cctest" performs the Cochrane’s Q test for the equality of more than two groups whose values are strictly binary (0 or 1). This is a generalisation of the McNemar’s test in the multi-sample case. The "block.anova" is the ANOVA with blocking, randomised complete block design (RCBD). In this case, for every combination of the block and treatment values, there is only one observation. The mathematics are the same as in the case of "twoway.anova", but the assumptions different and the testing procedure also different. In addition, no interaction is present.

Value

A vector with the test statistic and the p-value of each test. For the case of t-test, an extra column with the degrees of freedom is given. For the two way ANOVA there can can be either 2 or three F test statistics and hence the same number of p-values.

Author(s)

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

References


See Also

ttests, ftests

Examples

```r
x <- rnorm(200)
ing <- rbinom(200, 3, 0.5) + 1
res<-.anova(x, ina)
res<-.ftest(x, ina)
ing <- rbinom(200, 1, 0.5) + 1
x1 <- x[ ina == 1 ] ; x2 <- x[ ina == 2 ]
res<-.ttest2(x1, x2)
res<-.var2test(x1, x2)
```
Multinomial regression

## RCBD example 4.1 from Montgomery (2001), page 131-132

```r
x <- c(9.3, 9.4, 9.2, 9.7, 9.4, 9.3, 9.4, 9.6, 9.6, 9.8, 9.5, 10,
       10, 9, 9.7, 10.2)
tr <- rep(1:4, 4)
bl <- rep(1:4, each = 4)
res<-block.anova(x, tr, bl)
```

Multinomial regression

### Description

Multinomial regression.

### Usage

```r
multinom.reg(y, x, tol = 1e-07, maxiters = 50)
```

### Arguments

- **y**: The response variable. A numerical or a factor type vector.
- **x**: A matrix or a data.frame with the predictor variables.
- **tol**: This tolerance value to terminate the Newton-Raphson algorithm.
- **maxiters**: The maximum number of iterations Newton-Raphson will perform.

### Value

A list including:

- **iters**: The number of iterations required by the Newton-Raphson.
- **loglik**: The value of the maximised log-likelihood.
- **be**: A matrix with the estimated regression coefficients.

### Author(s)

Michail Tsagris

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

### References


### See Also

- `glm_logistic`
- `score.multinomregs`
- `logistic_only`
Multivariate kurtosis

Examples

```r
## Not run:
y <- iris[, 5]
x <- matrnorm(150, 3)
res <- multinom.reg(y, x)
## End(Not run)
```

## Description

Multivariate kurtosis.

## Usage

`mvkurtosis(x)`

## Arguments

`x`
A numerical matrix.

## Details

The multivariate kurtosis is calculated.

## Value

A number, the multivariate kurtosis.

## Author(s)

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

## References


## See Also

`colskewness, skew.test2, colmeans, colVars, colMedians`

## Examples

```r
x <- as.matrix(iris[, 1:4])
res <- mvkurtosis(x)
```
Multivariate Laplace random values simulation

Description
Multivariate Laplace random values simulation.

Usage
\[ \text{rmvlaplace}(n, \lambda, \mu, G, \text{seed} = \text{NULL}) \]

Arguments
\begin{itemize}
  \item \textbf{n} \hspace{1cm} \text{The sample size, a numerical value.}
  \item \textbf{\lambda} \hspace{1cm} \text{The parameter of the exponential distribution, a positive number.}
  \item \textbf{\mu} \hspace{1cm} \text{The mean vector.}
  \item \textbf{G} \hspace{1cm} \text{A \(d \times d\) covariance matrix with determinant 1.}
  \item \textbf{seed} \hspace{1cm} \text{If you want the same to be generated again use a seed for the generator, an integer number.}
\end{itemize}

Details
The algorithm uses univariate normal random values and transforms them to multivariate via a spectral decomposition.

Value
A matrix with the simulated data.

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

References

See Also
\texttt{rmvnorm, racg, rmvt}
Examples
m <- colmeans(as.matrix(iris[, 1:4]))
s <- cov(iris[,1:4])
s <- s / det(s)^0.25
lam <- 3
x <- rmvlaplace(100, lam, m, s)

Description
Multivariate normal and t random values simulation.

Usage
rmvnorm(n, mu, sigma, seed = NULL)
rmvt(n, mu, sigma, v, seed = NULL)

Arguments
- n: The sample size, a numerical value.
- mu: The mean vector in $R^d$.
- sigma: The covariance matrix in $R^d$.
- v: The degrees of freedom.
- seed: If you want the same to be generated again use a seed for the generator, an integer number.

Details
The algorithm uses univariate normal random values and transforms them to multivariate via a spectral decomposition. It is faster than the command "mvrnorm" available from MASS, and it allows for singular covariance matrices.

Value
A matrix with the simulated data.

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

References
Naive Bayes classifiers

See Also

gac, rmvlaplace, rmvt

Examples

```r
x <- as.matrix(iris[, 1:4])
m <- colmeans(x)
s <- cov(x)
y <- rmvnorm(1000, m, s)
res <- colmeans(y)
res <- cov(y)
y <- NULL
```

Naive Bayes classifiers

Gaussian, Poisson, geometric and multinomial naive Bayes classifiers.

Usage

```r
gaussian.nb(xnew = NULL, x, ina, parallel = FALSE)
poisson.nb(xnew, x, ina)
multinom.nb(xnew, x, ina)
geom.nb(xnew, x, ina, type = 1)
gammanb(xnew = NULL, x, ina, tol = 1e-07)
```

Arguments

- `xnew`: A numerical matrix with new predictor variables whose group is to be predicted. For the Gaussian naive Bayes, this is set to NULL, as you might want just the model and not to predict the membership of new observations. For the Gaussian case this contains any numbers, but for the multinomial and Poisson cases, the matrix must contain integer valued numbers only.

- `x`: A numerical matrix with the observed predictor variable values. For the Gaussian case this contains any numbers, but for the multinomial and Poisson cases, the matrix must contain integer valued numbers only.

- `ina`: A numerical vector with strictly positive numbers, i.e. 1, 2, 3 indicating the groups of the dataset. Alternatively this can be a factor variable.

- `type`: Type 1 refers to the case where the minimum is zero and type 2 for the case of the minimum being 1. This is for the geometric distribution. This argument is for the geometric distribution. Type 1 refers to the case where the minimum is zero and type 2 for the case of the minimum being 1.

- `tol`: The tolerance value to terminate the Newton-Raphson algorithm in the gamma distribution.

- `parallel`: If you want parallel computations set this equal to TRUE.
Naive Bayes classifiers

Value

For the Poisson and Multinomial naive Bayes classifiers the estimated group, a numerical vector with 1, 2, 3 and so on. For the Gaussian naive Bayes classifier a list including:

- **mu** A matrix with the mean vector of each group based on the dataset.
- **sigma** A matrix with the variance of each group and variable based on the dataset.
- **ni** The sample size of each group in the dataset.
- **est** The estimated group of the xnew observations. It returns a numerical value back regardless of the target variable being numerical as well or factor. Hence, it is suggested that you do `"as.numeric(target)"` in order to see what is the predicted class of the new data.

For the Gamma classifier a list including:

- **a** A matrix with the shape parameters.
- **b** A matrix with the scale parameters.
- **est** The estimated group of the xnew observations. It returns a numerical value back regardless of the target variable being numerical as well or factor. Hence, it is suggested that you do `"as.numeric(target)"` in order to see what is the predicted class of the new data.

Author(s)

Michail Tsagris

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

See Also

`gaussiannb.pred`, `colmeans`, `colVars`

Examples

```r
x <- as.matrix(iris[, 1:4])
a <- gaussian.nb(x, x, iris[, 5])
x1 <- matrix( rpois(100 * 4, 5), ncol = 4)
x2 <- matrix(rpois(50 * 4, 10), ncol = 4)
x <- rbind(x1, x2)
ina <- c(rep(1, 100), rep(2, 50))
res<-poisson.nb(x, x, ina)
res<-geom.nb(x, x, ina)
res<-multinom.nb(x, x, ina)
```
Natural Logarithm each element of a matrix

Usage

Log(x, na.rm = FALSE)

Arguments

x A matrix with data.

na.rm A boolean value (TRUE/FALSE) for removing NA.

Details

The argument must be a matrix. For vector the time was the same as R’s "log" function so we did not add it.

Value

A matrix where each element is the natural logarithm of the given argument.

Author(s)

Manos Papadakis

R implementation and documentation: Manos Papadakis <papadakm95@gmail.com>.

See Also

Lbeta,Lchoose,Choose

Examples

x <- matrix( runif( 100 * 100 ), ncol = 100 )
a <- log(x)
b <- Log(x)
all.equal(a, b) # true

x<-a<-b<-NULL
Natural logarithm of the beta function

Description

Natural logarithm of the beta function.

Usage

Lbeta(x, y)

Arguments

x
A numerical matrix, or a vector or just a number with positive numbers in either case.

y
A numerical matrix, or a vector or just a number with positive numbers in either case. The dimensions of y must match those of x.

Details

The function is faster than R’s lbeta when the dimensions of x any are large. If you have only two numbers, then lbeta is faster. But if you have for example two vectors of 1000 values each, Lbeta becomes two times faster than lbeta.

Value

The matrix, vector or number with the resulting values.

Author(s)

Manos Papadakis

R implementation and documentation: Manos Papadakis <papadakm95@gmail.com>.

References


See Also

Lgamma, beta.mle, diri.nr2
**Examples**

```r
x <- rexp(1000)
y <- rexp(1000)
a1 <- Lbeta(x, y)
```

---

**Description**

Natural logarithm of the gamma function and its derivatives.

**Usage**

- `Lgamma(x)`
- `Digamma(x)`
- `Trigamma(x)`

**Arguments**

- `x` A numerical matrix or vector with positive numbers in either case.

**Details**

We have spotted that the time savings come when there are more than 50 elements, with vector or matrix.

**Value**

The matrix or the vector with the resulting values.

**Author(s)**

Manos Papadakis

R implementation and documentation: Manos Papadakis <papadakm95@gmail.com>.

**References**


**See Also**

- `beta.mle`
- `diri.nr2`
Examples

```r
x <- matrix( rnorm(500 * 500), ncol = 500 )
a1 <- Lgamma(x)
a2 <- lgamma(x)
all.equal(as.vector(a1), as.vector(a2))

a1 <- Digamma(x)
a2 <- digamma(x)
all.equal(as.vector(a1), as.vector(a2))

x<-a1<-a2<-NULL
```

Description

Norm of a matrix.

Usage

```r
Norm(x, type = "F")
```

Arguments

- `x`: A matrix with numbers.
- `type`: The type of norm to be calculated. The default is "F" standing for Frobenius norm ("F" in R’s norm). The other options are "C" standing for the one norm ("o" in R’s norm), "R" for the identity norm ("I" in R’s norm) and "M" for the maximum modulus among elements of a matrix ("M" in R’s norm).

Value

A number, the norm of the matrix.

Author(s)

Manos Papadakis

R implementation and documentation: Manos Papadakis <papadakm95@gmail.com>.

See Also

`Dist,dista,colmeans`
Number of equal columns between two matrices

Examples

```r
x <- matrix( rnorm(10 * 10), ncol = 10 )
res<-Norm(x, "F")
res<-norm(x, "F")
res<-Norm(x, "M")
res<-norm(x, "M")
```

Description

Number of equal columns between two matrices.

Usage

```r
mat.mat(x, y)
```

Arguments

- `x`: A numerical matrix. See details for more information. It must have the same number of rows as `y`.
- `y`: A numerical matrix. See details for more information. It must have the same number of rows as `x`.

Details

The function takes each column of `x` and checks the number of times it matches a column of `y`. In the example below, we take the first 3 columns of iris as the `x` matrix. The `y` matrix is the whole of iris. We will see how many times, each column of `x` appears in the `y` matrix. The answer is 1 for each column.

Value

A numerical vector of size equal to the number of columns of `x`.

Author(s)

Manos Papadakis

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

See Also

`Match`, `colmeans`, `colMedians`
Examples

```r
x <- as.matrix(iris[, 1:3])
y <- iris
y[, 5] <- as.numeric(y[, 5])
y <- as.matrix(y)
res <- mat.mat(x, y)
```

Odds ratio and relative risk

**Description**

Odds ratio and relative risk.

**Usage**

```r
odds.ratio(x, a = 0.05, logged = FALSE)
rel.risk(x, a = 0.05, logged = FALSE)
```

**Arguments**

- `x`: A 2 x 2 matrix or a vector with 4 elements. In the case of the vector make sure it corresponds to the correct table.
- `a`: The significance level, set to 0.05 by default.
- `logged`: Should the p-values be returned (FALSE) or their logarithm (TRUE)?

**Details**

The odds ratio and the confidence interval are calculated.

**Value**

A list including:

- `res`: The estimated odds ratio and the p-value for the null hypothesis test that it is equal to 1.
- `ci`: The (1-a)% confidence interval for the true value of the odds ratio.

**Author(s)**

Michail Tsagris

R implementation and documentation: Michail Tsagris <mtsagris@uoc.gr>.
One sample t-test for a vector

References


See Also

odds, g2Test

Examples

x <- rpois(4, 30)+2
res<-odds.ratio(x)
res<-odds.ratio( matrix(x, ncol = 2) )

Description

One sample t-test for a vector.

Usage

ttest1(x, m, alternative = "unequal", logged = FALSE, conf = NULL)

Arguments

x  A numerical vector with the data.
m  The mean value under the null hypothesis.
alternative  The alternative hypothesis, "unequal", "greater" or "less".
logged  Should the p-values be returned (FALSE) or their logarithm (TRUE)?
conf  If you want a confidence interval supply the confidence level.

Details

The usual one sample t-test is implemented, only faster.

Value

A list including:

res  A two valued vector with the test statistic and its (logged) p-value.
ci  In the case you supplied a number in the input argument "conf" the relevant confidence interval will be returned as well.
Operations between two matrices or matrix and vector

**Author(s)**

Michail Tsagris

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

**See Also**

`ttest`, `anova1`, `ttests`

**Examples**

```r
x = rnorm(500)
res<-t.test(x, mu = 0)
res<-ttest1(x, 0, conf = 0.95)
```

**Description**

Operations between two matrices or matrix and vector.

**Usage**

```r
XopY.sum(x, y = NULL, oper = "*")
eachrow(x,y,oper = ",",method = NULL)
eachcol.apply(x,y,indices = NULL,oper = "",apply = "sum")
```

**Arguments**

- `x` A numerical matrix.
- `y` A second numerical matrix for "XopY.sum" whose dimensions must match the ones of `x`, or vector for "eachrow","eachcol.apply" whose length must match with the rows of `x`.
- `oper` The operation to be performed, either "+", "/", ",", "," or ".==".
- `method` A character value for choosing option to apply in the result. Options: 1) sum 2) max 3) min
  Does not work for `oper="=="`.
- `indices` An integer vector with indices to specific columns. Only for "eachcol.apply".
- `apply` A character value with the function to be applied in the columns of the matrix. Only for "eachcol.apply". Options: 1) sum 2) median 3) max 4) min
Details

XopY.sum: \( \sum(X \text{ op } Y) \) where \( \text{op} \) can be one of "+,-,*,/".

eachrow: \( X \text{ op } Y \) by row \text{ FUNCTION}(X \text{ op } Y) \) where "x" is matrix, "y" is vector with length as much as the columns of x and "op" is one of "+,-,*,/==", and "FUNCTION" is a specific method for applying in the result matrix (see argument method).

eachcol.apply: \text{ FUNCTION}(X \text{ op } Y) \text{ by column} \) where "x" is matrix, "y" is vector with length as much as the rows of x, "op" is one of "+,-,*,/" and "FUNCTION" is a specific method (see argument apply).

\text{NOTE}: Arguments "method" does not work for oper="==" and this operation works only in "eachrow".

Value

XopY.sum: \( \sum(X \text{ op } Y) \) where "op" can be one of "+,-,*,/".

eachrow: operation by row between a matrix and a vector."op" can be one of "+,-,*,/". If "suma=TRUE" then returns the sum of this operation.

eachcol.apply: operation by column between a matrix and a vector and applied a specific function."op" can be one of "+,-,*,/".

Author(s)

Manos Papadakis

R implementation and documentation: Manos Papadakis <papadakm95@gmail.com>.

See Also

Dist,dista,colmeans,Diag.fill,colMads,rowMads

Examples

```r
x <- matrix( rnorm(5 * 5), ncol = 5 )
y <- matrix( rnorm(5 * 5), ncol = 5 )
res<-XopY.sum(x, y, oper = "+")
y <- x[,1]
res<-eachrow(x,y)

all.equal(eachcol.apply(x,y),colsums(x*y))
```

```
x<-y<-NULL
```
Orthogonal matching pursuit variable selection

Description

Orthogonal matching pursuit variable selection.

Usage

ompr(y, x, ystand = TRUE, xstand = TRUE, method = "BIC", tol = 2 )
omp(y, x, xstand = TRUE, tol = qchisq(0.95, 1) + log(length(y)), type = "logistic")

Arguments

y The response variable, a numeric vector. For "ompr" this is a continuous variable. For "omp" this can be either a vector with discrete (count) data, 0 and 1, non negative values, strictly positive or proportions including 0 and 1.
x A matrix with the data, where the rows denote the observations and the columns are the variables.
ystand If this is TRUE the response variable is centered. The mean is subtracted from every value.
xstand If this is TRUE the independent variables are standardised.
method You can choose between the change in the BIC ("BIC"), the adjusted $R^2$ ("ar2"), the SSE ("SSE") or the classical p-value based ("pvalue").
tol The tolerance value to terminate the algorithm. This is the change in the criterion value between two successive steps. For "ompr" the default value is 2 because the default method is "BIC". For "omp" the default value is the 95% quantile of the $\chi^2$ distribution with 1 degree of freedom plus the logarithm of the sample size.
type This denotes the parametric model to be used each time. It depends upon the nature of y. The possible values are "logistic", "poisson", "quasipoisson", "quasibinomial", "normlog", "gamma", "weibull", "mv" (for multivariate response variable) or "multinomial".

Value

For "ompr" a list including:

runtime The runtime of the algorithm.
info A matrix with two columns. The selected variable(s) and the criterion value at every step.

For "omp" a list including:

runtime The runtime of the algorithm.
Outer function

phi

The $\phi$ parameter. In the cases of "quasipoisson", "quasibinomial" and "normlog" this is useful. For all other cases this is NULL.

info

A matrix with two columns. The selected variable(s) and the criterion value at every step.

Author(s)

Michail Tsagris

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

References


See Also

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

Examples

x <- matrnorm(100, 400)
y <- rnorm(100)
a <- ompr(y, x)
a
x <- NULL

Outer function

The outer function.

Usage

Outer(x, y, oper = "*")
Arguments

- **x**: A numerical vector.
- **y**: A numerical vector.
- **oper**: The available options are "\*" (multiplication), "/" (division), "+" (sum), "-" (subtraction), "^" (power raise), and "\n"

Details

The function is the same as R’s "outer", but works with vectors only and probably has less capabilities, but faster.

Value

A matrix with all the combinations.

Author(s)

Manos Papadakis and Michail Tsagris

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

See Also

`mat.mult, vecdist`

Examples

```r
x <- rnorm(10)
y <- rnorm(10)
res <- Outer(x, y)
```

Description

Permute the given vector.

Usage

```r
permutation(x, nperm = gamma(length(x)+1))
permutation.next(x, nperm = gamma(length(x)+1))
permutation.prev(x, nperm = gamma(length(x)+1))
bincomb(n)
```
Arguments

- **x**: A numeric vector with data.
- **nperm**: An integer value for returning specific number of combinations. By default, it is set to all combinations. Must be between $0 \leq nperm \leq \text{gamma(length(x)+1)}$.
- **n**: An integer value for the length of the binary number.

Details

This function implements "Permutation", which means all the possible combinations. In the permutation.next and permutation.prev if there aren't possible combinations it returns the same vector. "Binary Combinations" for "bincomb", means all the possible combinations for the binary number with length "n".

Value

Returns a matrix with all possible combinations of the given vector or a matrix row with one possible combination.

Author(s)

Manos Papadakis

R implementation and documentation: Manos Papadakis <papadakm95@gmail.com>

See Also

- `combn`, `comb_n`

Examples

```r
y <- rnorm(3)
b <- permutation(y)
b <- permutation.next(y)
b <- permutation.prev(y)
g <- bincomb(3)
```

Description

Permutation based p-value for the Pearson correlation coefficient.

Usage

`permcor(x, y, R = 999)`
Polyserial correlation

Arguments

- **x**  A numerical vector with the first variable.
- **y**  A numerical vector with the second variable.
- **R**  The number of permutations to be conducted; set to 999 by default.

Details

This is a very low computational calculation of the p-value. Try it yourselves.

Value

A vector consisting of two values, the Pearson correlation and the permutation based p-value.

Author(s)

Marios Dimitriadis and Michail Tsagris

R implementation and documentation: Marios Dimitriadis and Michail Tsagris <kmdimitriadis@gmail.com> and <mtsagris@csd.uoc.gr>

References


See Also

- **pc.skel**

Examples

```r
x <- iris[, 1]
y <- iris[, 2]
res<-permcor(x, y)
res<-permcor(x, y, R = 9999)
```

---

Polyserial correlation

Polyserial correlation

Description

Polyserial correlation.
Usage

poly.cor(x, y)

Arguments

x  The continuous variable.

y  The ordinal variable, a numeric vector with numbers starting from 1.

Details

The polyserial correlation between a continuous and an ordinal variable is calculated. The function is not super fast, yet is faster than other implementations we found.

Value

A list including:

est  A vector with the polyserial correlation and its estimated variance.

test  A vector with the test statistic and its associated p-value.

Author(s)

Michail Tsagris

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

References


See Also

correls, Table

Examples

x <- rnorm(100)
y <- rpois(100, 10) + 1
res<-poly.cor(x, y)
Pooled covariance matrix

Description

Pooled covariance matrix.

Usage

pooled.cov(x, ina)

Arguments

x A matrix with continuous data.
ina A numerical vector indicating the groups. The numbers must be consecutive and start from 1.

Details

The spatial median is at first computed (if not supplied) and then the covariance matrix.

Value

The spatial sign covariance matrix.

Author(s)

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

References


See Also

spat.med, spatmed.reg

Examples

res<-sscov( as.matrix(iris[, 1:4]) )
Prediction with some naive Bayes classifiers

Description
Prediction with some naive Bayes classifiers.

Usage

\[
\text{gaussiannb.pred}(xnew, m, s, ni) \\
\text{poissonnb.pred}(xnew, m) \\
\text{multinomnb.pred}(xnew, m) \\
\text{gammanb.pred}(xnew, a, b) \\
\text{geomnb.pred}(xnew, prob)
\]

Arguments

- **xnew**: A numerical matrix with new predictor variables whose group is to be predicted. For the Gaussian case this contains any numbers, but for the multinomial and Poisson cases, the matrix must contain integer valued numbers only.
- **m**: A matrix with the group means. Each row corresponds to a group.
- **s**: A matrix with the group column-wise variances. Each row corresponds to a group.
- **ni**: A vector with the frequencies of each group.
- **a**: A vector with the shape parameters of each group.
- **b**: A vector with the scale parameters of each group.
- **prob**: A vector with the probability parameters of each group.

Value
A numerical vector with 1, 2, ... denoting the predicted group.

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

See Also

- `gaussian.nb`, `colpoisson.mle`, `colVars`
Examples

```r
ina <- sample(1:150, 100)
x <- as.matrix(iris[, 1:4])
id <- as.numeric(iris[, 5])
a <- gaussian.nb(xnew = NULL, x[ina, ], id[ina])
est <- gaussiannb.pred(x[-ina, ], a$mu, a$sigma, a$ni)
res<-table(id[-ina], est)
```

Description

Quasi binomial regression for proportions.

Usage

```r
prop.reg(y, x, varb = "quasi", tol = 1e-09, maxiters = 100)
prop.regs(y, x, varb = "quasi", tol = 1e-09, logged = FALSE, maxiters = 100)
```

Arguments

- `y`: A numerical vector proportions. 0s and 1s are allowed.
- `x`: For the "prop.reg" a matrix with data, the predictor variables. This can be a matrix or a data frame. For the "prop.regs" this must be a numerical matrix, where each columns denotes a variable.
- `tol`: The tolerance value to terminate the Newton-Raphson algorithm. This is set to $10^{-9}$ by default.
- `varb`: The type of estimate to be used in order to estimate the covariance matrix of the regression coefficients. There are two options, either "quasi" (default value) or "glm". See the references for more information.
- `logged`: Should the p-values be returned (FALSE) or their logarithm (TRUE)?
- `maxiters`: The maximum number of iterations before the Newton-Raphson is terminated automatically.

Details

We are using the Newton-Raphson, but unlike R’s built-in function "glm" we do no checks and no extra calculations, or whatever. Simply the model. The "prop.regs" is to be used for very many univariate regressions. The "x" is a matrix in this case and the significance of each variable (column of the matrix) is tested. The function accepts binary responses as well (0 or 1).
Quasi binomial regression for proportions

Value

For the "prop.reg" function a list including:

- **iters**: The number of iterations required by the Newton-Raphson.
- **varb**: The covariance matrix of the regression coefficients.
- **phi**: The phi parameter is returned if the input argument "varb" was set to "glm", otherwise this is NULL.
- **info**: A table similar to the one produced by "glm" with the estimated regression coefficients, their standard error, Wald test statistic and p-values.

For the "prop.regs" a two-column matrix with the test statistics (Wald statistic) and the associated p-values (or their logarithm).

Author(s)

Michail Tsagris

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

References


See Also

anova_propreg univglms, score.glms, logistic_only

Examples

```r
## Not run:
y <- rbeta(100, 1, 4)
x <- matrix(rnorm(100 * 3), ncol = 3)
a <- prop.reg(y, x)
y <- rbeta(100, 1, 4)
x <- matrix(rnorm(400 * 100), ncol = 400)
b <- prop.regs(y, x)
res<mean(b[, 2] < 0.05)
## End(Not run)
```
Quasi Poisson regression for count data

Quasi Poisson regression

Description

Quasi Poisson regression.

Usage

qpois.reg(x, y, full = FALSE, tol = 1e-09, maxiters = 100)
qpois.regs(x, y, tol = 1e-09, logged = FALSE)

Arguments

x        For the "qpois.reg" a matrix with data, the predictor variables. This can be a matrix or a data frame. For the "qpois.regs" this must be a numerical matrix, where each columns denotes a variable.
y        A numerical vector with positive discrete data.
full        If this is FALSE, the coefficients, the deviance and the estimated phi parameter will be returned only. If this is TRUE, more information is returned.
tol        The tolerance value to terminate the Newton-Raphson algorithm. This is set to \(10^{-9}\) by default.
logged        Should the p-values be returned (FALSE) or their logarithm (TRUE)?
maxiters        The maximum number of iterations before the Newton-Raphson is terminated automatically.

Details

We are using the Newton-Raphson, but unlike R's built-in function "glm" we do no checks and no extra calculations, or whatever. Simply the model, unless the user requests for the Wald tests of the coefficients. The "qpois.regs" is to be used for very many univariate regressions. The "x" is a matrix in this case and the significance of each variable (column of the matrix) is tested.

Value

For the "prop.reg" a list including: When full is FALSE

be        The regression coefficients.
devi        The deviance of the model.
varb        The covariance matrix of the beta coefficients.
phi        The phi parameter, the estimate of dispersion.

When full is TRUE, the additional item is:
info The regression coefficients, their standard error, their Wald test statistic and their p-value.

For the "prop.regs" a two-column matrix with the test statistics (Wald statistic) and the associated p-values (or their logarithm).

Author(s)

Manos Papadakis and Marios Dimitriadis

R implementation and documentation: Manos Papadakis <papadakm95@gmail.com> and Marios Dimitriadis <kmdimitriadis@gmail.com>.

References


See Also

prop.reg univglms, score.glms, poisson_only

Examples

```r
## Not run:
y <- rbinom(100, 10, 0.6)
x <- matrix(rnorm(100*3), ncol = 3)
mod1 <- glm(y ~ x, quasipoisson)
res<-summary(mod1)
res<-qpois.reg(x, y, full = TRUE)
res<-qpois.regs(x, y)
## End(Not run)
```

Random intercepts linear mixed models

Random intercepts linear mixed models (for balanced data with a single identical covariate).

Usage

```r
rint.reg(y, x, id, tol = 1e-08, ranef = FALSE, maxiters = 100)
rint.regbx(y, x, id)
```
Arguments

y  A numerical vector with the data. The subject values.

x  For the case of "rint.reg" this can be a vector or a numerical matrix with data. In the case of "rint.regbx" this is a numerical vector with the same length as y indicating the fixed predictor variable. Its values are the same for all levels of y. An example of this x is time which is the same for all subjects.

id  A numerical variable with 1, 2, ... indicating the subject.

tol  The tolerance level to terminate the generalised elast squares algorithm.

ranef  If you want to obtain the random effects (random intercepts) set this equal to TRUE.

maxiters  The max number of iterations that can take place in a regression.

Details

Random intercepts linear mixed models with compound covariance structure is fitted in both functions. The "rint.reg" allows any numerical matrix, with balanced or unbalanced data. See Demidenko (2013, pg. 65-67) for more information.

The "rint.regbx" is a special case of a balanced random intercepts model with a compound symmetric covariance matrix and one single covariate which is constant for all replicates. An example, is time, which is the same for all subjects. Maximum likelihood estimation has been performed. In this case the mathematics exist in a closed formula (Demidenko, 2013, pg. 67-69).

Value

A list including:

info  A vector with the random intercepts variance (between), the variance of the errors (within), the log-likelihood, the deviance (twice the log-likelihood) and the BIC. In the case of "rint.reg" it also includes the number of iterations required by the generalised least squares.

be  The estimated regression coefficients, which in the case of "rint.regbx" are simply two: the constant and the slope (time effect).

ranef  The random intercepts effects.

Author(s)

Michail Tsagris

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

References

Random values simulation from a von Mises distribution

See Also

rm.lines, varcomps.mom, colvarcomps.mom

Examples

## Not run:
y <- rnorm(100)
x <- rnorm(10)
x <- rep(x, 10)
id <- rep(1:10, each = 10)
system.time( for (i in 1:40) a <- rint.reg(y, x, id) )
## End(Not run)

Random values simulation from a von Mises distribution

Description

It generates random vectors following the von Mises distribution. The data can be spherical or hyper-spherical.

Usage

rvonmises(n, m, k, rads = TRUE)

Arguments

n The sample size.
m The mean angle expressed in radians or degrees.
k The concentration parameter. If k is zero the sample will be generated from the uniform distribution over \((0, 2\pi)\).
rads If the mean angle is expressed in radians, this should be TRUE and FALSE otherwise. The simulated data will be expressed in radians or degrees depending on what the mean angle is expressed.

Details

The mean direction is transformed to the Euclidean coordinates (i.e. unit vector) and then the fvmf function is employed. It uses a rejection sampling as suggested by Andrew Wood in 1994. I have mentioned the description of the algorithm as I found it in Dhillon and Sra in 2003. Finally, the data are transformed to radians or degrees.

Value

A vector with the simulated data.
Ranks of the values of a vector

Author(s)
Michail Tsagris and Manos Papadakis
R implementation and documentation: Michail Tsagris <mtsagris@uoc.gr> and Manos Papadakis <papadakm85@gmail.com>

References

See Also
vm.mle, rvmf

Examples
x <- rvmises(1000, 2, 25, rads = TRUE)
res<-vm.mle(x)

Ranks of the values of a vector

Description
Ranks of the values of a vector.

Usage
Rank(x, method = "average", descending = FALSE)

Arguments

- x: A numerical vector with data.
- method: A character string for choosing method. Must be one of "average", "min", "max".
- descending: A boolean value (TRUE/FALSE) for sorting the vector in descending order. By default sorts the vector in ascending.

Details
The ranks of the values are returned, the same job as "rank". If you want you can choose descending/ascending order for all methods.
Value

A vector with the ranks of the values.

Author(s)

Manos Papadakis

R implementation and documentation: Manos Papadakis <papadakm95@gmail.com>.

See Also

colRanks, correls

Examples

```r
x <- rnorm(100)
a1 <- Rank(x)
a2 <- rank(x)
```

Description

Reading the files of a directory.

Usage

```r
read.directory(path.directory)
read.examples(path.man)
```

Arguments

- `path.directory`: The full path to the directory. For example: `"C:\User\name\Documents\R\Rfast_1.8.0\"
- `path.man`: The full path to the directory with the Rd files in it. For example: `"C:\User\name\Documents\R\Rfast_1.8.0\man\"

Details

For function `read.directory()`: Takes as an argument a full path to a directory and returns the names of the files.

For function `read.examples()`: Takes as an argument a full path to the directory of the Rd files. If you don’t want the program to read any file add at the top of the file the attribute "%[dont read]".
Repeated measures anova

Description
Repeated measures anova.

Usage
rm.anova(y, logged = FALSE)

Arguments
y A matrix with the data, where each column refers to a different measurement. The rows denote the subjects.
logged Should the p-values be returned (FALSE) or their logarithm (TRUE)?
Description

Replicate columns/rows.

Usage

rep_col(x, n)
rep_row(x, n)
Arguments

- **x**: A vector with data.
- **n**: Number of new columns/rows.

Value

A matrix where each column/row is equal to "x".

Author(s)

Manos Papadakis
R implementation and documentation: Manos Papadakis <papadakm95@gmail.com>.

See Also

rowMins, rowFalse, nth, colrange, colMedians, colVars, colSort, rowSort, rowTrue

Examples

```r
x <- runif(10)
all.equal(rep_col(x, 10), matrix(x, nrow=length(x), ncol=10))
all.equal(rep_row(x, 10), matrix(x, ncol=length(x), nrow=10, byrow=TRUE))
```

Description

Represantation of Stack.

Usage

Stack(x, type=NULL)

Arguments

- **x**: Any type that could be convert to vector or an integer value.
- **type**: A type for the Stack, "integer", "numeric" or any other that accepts one argument.

Details

Stack is an abstract data type - data structure based on the principle of last in first out. To access the 3 fields, use operator "$".
Round each element of a matrix/vector

Value

An object of class "Stack". This object holds 3 fields:
pop: remove the first element (from the top). top: access the first element (from the top). push: add an element to the top of the Stack.

Author(s)

R implementation and documentation: Manos Papadakis <papadakm95@gmail.com>.

See Also

colShuffle, colVars, colmeans, read.directory

Examples

x<-Stack(10,type=integer)
  x$push(5)
  x$push(10)
  x$top() == 10
  x$pop()
  x$top() == 5

y<-rnorm(10)
x<-Stack(x)
  x$push(5) # length increased to 11
  x$top() # access the last element that pushed, 5
  x$pop() # pop the last element that pushed

Description

Round each element of a matrix/vector.

Usage

Round(x,digit=0,na.rm = FALSE)

Arguments

x A numeric matrix/vector with data or NA. NOT integer values.
digit An integer value for 0...N-1 where N is the number of the digits. By default is 0.
na.rm TRUE or FALSE for remove NAs if exists.
Row - Wise matrix/vector count the frequency of a value

Details
Round is a very fast C++ implementation. Especially for large data. It handles NA.

Value
A vector/matrix where each element is been rounded in the given digit.

Author(s)
Manos Papadakis
R implementation and documentation: Manos Papadakis <papadakm95@gmail.com>.

See Also
Lchoose, Log, Choose

Examples
```r
x <- matrix(rnorm(500 * 100), ncol = 100)
system.time(a <- Round(x, 5))
system.time(b <- round(x, 5))
all.equal(a, b) # true
x <- rnorm(1000)
system.time(a <- Round(x, 5))
system.time(b <- round(x, 5))
all.equal(a, b) # true
```

Description
Row - Wise matrix/vector count the frequency of a value.

Usage
```r
count_value(x, value)
colCountValues(x, values, parallel = FALSE)
rowCountValues(x, values, parallel = FALSE)
```

Arguments
- `x`: A vector with the data (numeric or character) or a numeric matrix.
- `value`: The value, numeric or character, to check its frequency in the vector "x".
- `values`: a vector with the values to check its frequency in the matrix "x" by row or column.
- `parallel`: Do you want to do it in parallel in C++? TRUE or FALSE. Works with every other argument.
Row-wise minimum and maximum

Details

The functions is written in C++ in order to be as fast as possible. The "x" and "value" must have the same type. The type can be numeric or character.

Value

The frequency of a value/values in a vector in linear time or by row/column in a matrix.

Author(s)

Manos Papadakis

R implementation and documentation: Manos Papadakis <papadakm95@gmail.com>.

See Also

Median, binary_search, Order, nth

Examples

```r
x <- rnorm(100)
value <- x[50]
system.time( count_value(x,value) )
y <- sample(letters,replace=TRUE)
value <- "r"
system.time( count_value(y,value) )
values <- sample(x,100,replace=TRUE)
x <- matrix(x,100,100)
res<-colCountValues(x,values)
res<-rowCountValues(x,values)
x<-value<-values<-y<-NULL
```

Row-wise minimum and maximum

Row-wise minimum and maximum of a matrix.

Description

Row-wise minimum and maximum of a matrix.

Usage

```r
rowMins(x, value = FALSE)
rowMaxs(x, value = FALSE)
rowMinsMaxs(x)
```
**Row-wise true value**

**Arguments**

- **x**
  - A numerical matrix with data.
- **value**
  - If the value is FALSE it returns the indices of the minimum/maximum, otherwise it returns the minimum and maximum values.

**Value**

A vector with the relevant values.

**Author(s)**

Manos Papadakis

R implementation and documentation: Manos Papadakis <papadakm95@gmail.com>.

**See Also**

colMins, colMaxs, nth, rowrange, colMedians, colVars, colSort, rowSort

**Examples**

```r
x <- matrix( rnorm(500 * 500), ncol = 500 )

system.time( s1 <- rowMins(x) )
system.time( s2 <- apply(x, 1, min) )

system.time( s1 <- rowMaxs(x) )
system.time( s2 <- apply(x, 1, max) )

system.time( s1 <- c(apply(x, 1, min),apply(x, 1, max)) )
system.time( s2 <- rowMinsMaxs(x) )

x<-s1<-s2<-NULL
```

---

**Description**

Row-wise true value of a matrix.

**Usage**

- `rowTrue(x)`
- `rowFalse(x)`
- `rowTrueFalse(x)`
Search for variables with zero range in a matrix

Arguments

x A logical matrix with data.

Value

An integer vector where item "i" is the number of the true/false values of "i" row.

Author(s)

Manos Papadakis

R implementation and documentation: Manos Papadakis <papadakm95@gmail.com>.

See Also

rowMins, colFalse, nth, rowrange, rowMedians, rowVars, colTrue

Examples

x <- matrix(as.logical(rbinom(100*100,1,0.5)),100,100)

s1 <- rowTrue(x)

s1 <- rowFalse(x)

s1 <- rowTrueFalse(x)

x<-s1<-NULL

---

Search for variables with zero range in a matrix

Search for variables with zero range in a matrix

---

Description

Search for variables with zero range in a matrix.

Usage

check_data(x, ina = NULL)

Arguments

x A matrix or a data.frame with the data, where rows denotes the observations and the columns contain the dependent variables.

ina If your data are grouped, for example there is a factor or numerical variable indicating the groups of the data supply it here, otherwise leave it NULL.
Significance testing for the coefficients of Quasi binomial or the quasi Poisson regression

Details

The function identifies the variables with zero range, instead of a zero variance as this is faster. It will work with matrices and data.frames.

Value

A numerical vector of length zero if no zero ranged variable exists, or of length at least one with the index (or indices) of the variable(s) that need attention or need to be removed.

Author(s)

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

See Also

colrange, colVars

Examples

```r
x <- matrix(rnorm(100 * 100), ncol = 100)
res <- check_data(x)

## some variables have a constant value
x[, c(1, 10, 50, 70)] <- 1
res <- check_data(x)

id <- rep(1:4, each = 25)
x[1:25, 2] <- 0
res <- check_data(x)  ## did not use the id variable
res <- check_data(x, id)  ## see now
x <- NULL
```

Significance testing for the coefficients of Quasi binomial or the quasi Poisson regression

```
Significance testing for the coefficients of Quasi binomial or the quasi Poisson regression
```

Description

Significance testing for the coefficients of Quasi binomial or the quasi Poisson regression.

Usage

```r
anova_propreg(mod, poia = NULL)
anova_qpois.reg(mod, poia = NULL)
```
Significance testing for the coefficients of Quasi binomial or the quasi Poisson regression

Arguments

mod       An object as returned by the "prop.reg" or the "qpois.reg" function.
poi     If you want to test the significance of a single coefficient this must be a number. In this case, the "prop.reg" or the "qpois.reg" function contains this information. If you want more coefficients to be tested simultaneously, e.g. for a categorical predictor, then this must contain the positions of the coefficients. If you want to see if all coefficients are zero, like an overall F-test, leave this NULL.

Details

Even though the name of this function starts with anova it is not an ANOVA type significance testing, but a Wald type.

Value

A vector with three elements, the test statistic value, its associated p-value and the relevant degrees of freedom.

Author(s)

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

References


See Also

prop.reg,qpois.reg,univglms,score.glms,logistic_only

Examples

## Not run:
y <- rbeta(1000, 1, 4)
x <- matrix(rnorm(1000 * 3), ncol = 3)
a <- prop.reg(y, x)
## all coefficients are tested
res<-anova_propreg(a)
## the first predictor variable is tested
res<-anova_propreg(a, 2)
a ## this information is already included in the model output
## the first and the second predictor variables are tested
res<-anova_propreg(a, 2:3)

## End(Not run)
Simulation of random values from a Bingham distribution

Simulating from a Bingham distribution

Description

Simulation from a Bingham distribution using the code suggested by Kent et al. (2013).

Usage

rbing(n, lam)

Arguments

n  Sample size.

lam  Eigenvalues of the diagonal symmetric matrix of the Bingham distribution. See details for more information on this.

Details

The user must have calculated the eigenvalues of the diagonal symmetric matrix of the Bingham distribution. The function accepts the q-1 eigenvalues only. This means, that the user must have subtracted the lowest eigenvalue from the rest and give the non zero ones. The function uses rejection sampling.

Value

A matrix with the simulated data.

Author(s)

Michail Tsagris

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

References


See Also

rvmf
Examples

```r
x <- rbing( 100, c(1, 0.6, 0.1) )
x
```

Description

Simulation of random values from a Bingham distribution with any symmetric matrix.

Usage

```r
rbingham(n, A)
```

Arguments

- `n`: Sample size.
- `A`: A symmetric matrix.

Details

The eigenvalues of the $q \times q$ symmetric matrix $A$ are calculated and the smallest of them is subtracted from the rest. The $q - 1$ non-zero eigenvalues are then passed to `rbing`. The generated data are then right multiplied by $V^T$, where $V$ is the matrix of eigenvectors of the matrix $A$.

Value

A matrix with the simulated data.

Author(s)

Michail Tsagris

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

References


Simulation of random values from a normal distribution

See Also

rvmf

Examples

```r
A <- cov( iris[, 1:4] )
x <- rbingham(100, A)
x
```

Description

Simulation of random values from a normal distribution.

Usage

`Rnorm(n, m = 0, s = 1, seed = NULL)`

Arguments

- `n`: The sample size.
- `m`: The mean, set to 0 by default.
- `s`: The standard deviation, set to 1 by default.
- `seed`: If you want the same to be generated again use a seed for the generator, an integer number.

Details

By using the Ziggurat method of generating standard normal variates, this function is really fast when you want to generate large vectors. For less than 2,000 this might make no difference when compared with R’s `rnorm`, but for 10,000 this will be 6-7 times faster.

Value

A vector with `n` values.

Author(s)

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

See Also

`matrnorm`, `rvonmises`, `rvmf`, `rmvnorm`
Examples

```r
x <- Rnorm(500)
```

Description

It generates random vectors following the von Mises-Fisher distribution. The data can be spherical or hyper-spherical.

Usage

```r
rvmf(n, mu, k)
```

Arguments

- `n` The sample size.
- `mu` The mean direction, a unit vector.
- `k` The concentration parameter. If `k = 0`, random values from the spherical uniform will be drawn. Values from a multivariate normal distribution with zero mean vector and the identity matrix as the covariance matrix. Then each vector becomes a unit vector.

Details

It uses a rejection sampling as suggested by Andrew Wood (1994).

Value

A matrix with the simulated data.

Author(s)

Michail Tsagris and Manos Papadakis

R implementation and documentation: Michail Tsagris <mtsagris@uoc.gr> and Manos Papadakis <papadakm85@gmail.com>

References


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.

Usage

pc.skel(dataset, method = "pearson", alpha = 0.01, R = 1, stat = NULL, ini.pvalue = NULL)

Arguments

dataset A numerical matrix with the variables. If you have a data.frame (i.e. categorical data) turn them into a matrix using data.frame.to_matrix. Note, that for the categorical case data, the numbers must start from 0. No missing data are allowed.

method If you have continuous data, you can choose either "pearson" or "spearman". If you have categorical data though, this must be "cat". In this case, make sure the minimum value of each variable is zero. The g2Test and the relevant functions work that way.

alpha The significance level (suitable values in (0, 1)) for assessing the p-values. Default (preferred) value is 0.01.

R The number of permutations to be conducted. The p-values are assessed via permutations. Use the default value if you want no permutation based assessment.

stat If the initial test statistics (univariate associations) are available, pass them through this parameter.

ini.pvalue if the initial p-values of the univariate associations are available, pass them through this parameter.

See Also

vmf.mle, rvonmises, iag.mle

Examples

```r
m <- rnorm(4)
m <- m/sqrt(sum(m^2))
x <- rvmf(1000, m, 25)
m
res<-vmf.mle(x)
```
Skeleton of the PC algorithm

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 stage 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. If the unlikely, but not impossible, event of all p-values being the same, the test statistic divided by the degrees of freedom is used as a means of choosing which conditioning set is to be used first.

If two or more p-values are below the machine epsilon (.Machine$double.eps which is equal to 2.220446e-16), all of them are set to 0. To make the comparison or the ordering feasible we use the logarithm of 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) with no permutations, 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 according to Tsamardinos et al. (2006). We have modified it so that it calculates the p-value using permutations.

Value

A list including:

- `stat`: The test statistics of the univariate associations.
- `ini.pvalue`: The initial p-values univariate associations.
- `pvalue`: The logarithm of the p-values of the univariate associations.
- `runtime`: The amount of time it took to run the algorithm.
- `kappa`: The maximum value of k, the maximum cardinality of the conditioning set at which the algorithm stopped.
- `n.tests`: The number of tests conducted during each k.
- `G`: The adjancency matrix. A value of 1 in G[i, j] appears in G[j, i] also, indicating that i and j have an edge between them.
- `sepset`: A list with the separating sets for every value of k.

Author(s)

Marios Dimitriadis.

R implementation and documentation: Marios Dimitriadis <kmdimitriadis@gmail.com>.

References

Skewness and kurtosis coefficients


See Also

g2Test, g2Test_univariate, cora, correls

Examples

# simulate a dataset with continuous data
dataset <- matrix(rnorm(100 * 50, 1, 100), nrow = 100)
a <- pc.skel(dataset, method = "pearson", alpha = 0.05)

Skewness and kurtosis coefficients

**Skewness and kurtosis coefficients**

Description

Skewness and kurtosis coefficients.

Usage

skew(x, pvalue = FALSE)
kurt(x, pvalue = FALSE)

Arguments

- **x**: A numerical vector with data.
- **pvalue**: If you want a hypothesis test that the skewness or kurtosis are significant set this to TRUE. This checks whether the skewness is significantly different from 0 and whether the kurtosis is significantly different from 3.

Details

The sample skewness and kurtosis coefficient are calculated. For the kurtosis we do not subtract 3.

Value

If "pvalue" is FALSE (default value) the skewness or kurtosis coefficients are returned. Otherwise, the p-value of the significance of the coefficient is returned.
Some summary statistics of a vector for each level of a grouping variable

Author(s)

Klio Lakiotaki
R implementation and documentation: Klio Lakiotaki <kliolak@gmail.com>.

References

https://en.wikipedia.org/wiki/Skewness
https://en.wikipedia.org/wiki/Kurtosis

See Also
colskewness, skew.test2, colmeans, colVars, colMedians

Examples

```r
x <- rgamma(500,1, 4)
res<-skew(x)
res<-kurt(x, TRUE)
```

---

Some summary statistics of a vector for each level of a grouping variable

**Description**

Some summary statistics of a vector for each level of a grouping variable.

**Usage**

```r
group(x, ina, method="sum", ina.min=NULL, ina.max = NULL, ina.length.unique=NULL, mad.method="median")
group.sum(x, ina, ina.max = NULL, ina.min = NULL)
group.mean(x, ina, ina.max = max(ina))
```

**Arguments**

- `x` A numerical vector with data.
- `ina` A numerical vector with numbers. Note that zero and negative values are not allowed as this can cause R to run forever or crash.
- `ina.length.unique` Length of the unique numerical values of ina argument.
- `method` A character vector with values "sum", "var", "all", "any", "mad", "mean", "med", "min", "max", "min.max".
- `ina.max` Maximum number for vector ina.
- `ina.min` Minimum number for vector ina.
- `mad.method` A character vector with values "median", for median absolute deviation or "mean", for mean absolute deviation. This works only with method="mad".
Details

This command works only for vectors. Median absolute deviation, mean, median, minimum, maximum are some of the options offered.

Value

A vector with the variance, or standard deviation, or mean, or minimum, or maximum, or median, or minimum-maximum of x for each distinct value of ina.

Author(s)

Manos Papadakis and Michail Tsagris

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

See Also

colmeans, colVars, colMedians

Examples

```r
## Not run:
x <- rgamma(100,1, 4)
inga <- sample(1:5, 100, TRUE)
res<-group(x, ina,method="var")

## End(Not run)
```

Description

Fast sorting a vector.

Usage

Sort(x, descending=FALSE, partial=NULL, stable=FALSE, na.last=NULL)
Sort.int(x)
sort_cor_vectors(x, base, stable = FALSE, descending = FALSE)
Sort - Integer Sort - Sort a vector corresponding to another

Arguments

- **x**: A numerical/integer/character vector with data.
- **base**: A numerical/character vector to help sorting the x.
- **descending**: A boolean value (TRUE/FALSE) for sorting the vector in descending order. By default sorts the vector in ascending.
- **partial**: This argument has two usages. The first is an index number for sorting partial the vector. The second is a vector with 2 values, start and end c(start,end). Gives you a vector where the elements between start and end will be sorted only. Not character vector.
- **stable**: A boolean value (TRUE/FALSE) for choosing a stable sort algorithm. Stable means that discriminates on the same elements. Not character vector.
- **na.last**: Accept 4 values. TRUE, FALSE, NA, NULL. TRUE/FALSE: for put NAs last or first. NA: for remove NAs completely from vector. NULL: by default. Leave it like that if there is no NA values.

Details

This function uses the sorting algorithm from C++. The implementation is very fast and highly optimised. Especially for large data.

Value

- **Sort and Sort.int**: The sorted vector.
- **sort_cor_vectors**: The first argument but sorted according to the second.

Author(s)

Manos Papadakis

R implementation and documentation: Manos Papadakis <papadakm95@gmail.com>.

See Also

nth, colnth, rownth, sort_unique, Round

Examples

```r
x <- rnorm(1000)
system.time( s1 <- Sort(x) )
system.time( s2 <- sort(x) )
all.equal(s1,s2) #true but not if many duplicates.

system.time( s1 <- Sort(x,partial=100) )
system.time( s2 <- sort(x,partial=100) )
all.equal(s1,s2) #true
```
system.time( s1 <- Sort(x,stable=TRUE) )
system.time( s2 <- sort(x) )
all.equal(s1,s2) #true

x <- as.character(x)
system.time( s1 <- Sort(x) )
system.time( s2 <- sort(x) )
all.equal(s1,s2) #true

y <- runif(1000)
b <- sort_cor_vectors(x,y)

x<-rpois(100,100)
all.equal(Sort.int(x),sort.int(x))

x<-y<-s1<-s2<-NULL

Sort and unique numbers

Description

Sort and unique numbers.

Usage

sort_unique(x)

Arguments

x A numeric vector.

Details

The "sort_unique" function implements R's "unique" function using C++'s function but also sort the result. The "sort_unique.length" returns the length of the unique numbers only for integers.

Value

Returns the discrete values but sorted or their length (depending on the function you do).

Author(s)

Manos Papadakis

R implementation and documentation: Manos Papadakis <papadakm95@gmail.com>
Sorting of the columns-rows of a matrix

See Also

colSort, rowSort, sort_cor_vectors

Examples

```r
y <- rnorm(100)
a <- sort_unique(y)
b <- sort.int(unique(y))
all.equal(as.vector(a),as.vector(b))
x <- rpois(1000,10)
sort_unique.length(x)
length(sort_unique(x))
```

x<-a<-b<-NULL

Description

Fast sorting of the columns-rows of a matrix.

Usage

```r
colSort(x, descending = FALSE, stable = FALSE, parallel = FALSE)
rowSort(x, descending = FALSE, stable = FALSE, parallel = FALSE)
sort_mat(x, by.row = FALSE, descending = FALSE, stable = FALSE, parallel = FALSE)
```

Arguments

- `x`: A numerical matrix with data.
- `descending`: If you want the sorting in descending order, set this to TRUE.
- `stable`: If you the stable version, so that the results are the same as R’s (in the case of ties) set this to TRUE. If this is TRUE, the algorithm is a bit slower.
- `parallel`: Do you want to do it in parallel in C++? TRUE or FALSE. Works with every other argument.
- `by.row`: TRUE or FALSE for applying sort in rows or column.

Value

The matrix with its columns-rows (or rows) independently sorted.

Author(s)

Manos Papadakis
R implementation and documentation: Manos Papadakis <papadakm95@gmail.com>.
See Also
nth, colMaxs, colMins, colrange, sort_cor_vectors, sort_unique

Examples

```r
x <- matrix(rnorm(100 * 500), ncol = 500)
system.time(s1 <- colSort(x))
system.time(s2 <- apply(x, 2, sort))
all.equal(as.vector(s1), as.vector(s2))

x <- NULL
```

Source many R files

Description

Source many R/Rd files.

Usage

```r
sourceR(path, local = FALSE, encode = "UTF-8", print.errors = FALSE)
sourceRd(path, print.errors = FALSE)
```

Arguments

- `path`: An full path to the directory where R file are.
- `local`: TRUE, FALSE or an environment, determining where the parsed expressions are evaluated. FALSE (the default) corresponds to the user’s workspace (the global environment) and TRUE to the environment from which source is called.
- `encode`: Character vector. The encoding(s) to be assumed when file is a character string: see file. A possible value is "unknown" when the encoding is guessed: see the "Encodings" section.
- `print.errors`: A boolean value (TRUE/FALSE) for printing the errors, if exists, for every file.

Details

Reads many R files and source them.

Value

Returns the files that had produced errors during source.

Author(s)

R implementation and documentation: Manos Papadakis <papadakm95@gmail.com>.
Spatial median for Euclidean data

See Also
read.directory, AddToNamespace

Examples

# for example: path="C:\some_file\R\" where is R files are
# system.time( a<-sourceR(path) )
# for example: path="C:\some_file\man\" where is Rd files are
# system.time( a<-sourceRd(path) )

Description

Spatial median for Euclidean data.

Usage

spat.med(x, tol = 1e-09)

Arguments

x A matrix with Euclidean data, continuous variables.

 tol A tolerance level to terminate the process. This is set to 1e-09 by default.

Details

The spatial median, using a fixed point iterative algorithm, for Euclidean data is calculated. It is a robust location estimate.

Value

A vector with the spatial median.

Author(s)

Manos Papadakis and Michail Tsagris
R implementation and documentation: Michail Tsagris <mtsagris@uoc.gr> and Manos Papadakis <papadakm95@gmail.com>
Spatial median regression

References


See Also

colMedians

Examples

```r
res<-spat.med( as.matrix( iris[, 1:4] ) )
res<-colMeans( as.matrix(iris[, 1:4]) )
res<-colMedians( as.matrix(iris[, 1:4]) )
```

Description

Spatial median regression with Euclidean data.

Usage

```
spatmed.reg(y, x, tol = 1e-07)
```

Arguments

- `y`: A matrix with the response variable.
- `x`: The predictor variable(s), they have to be continuous.
- `tol`: The threshold upon which to stop the iterations of the Newton-Rapshon algorithm.

Details

The objective function is the minimization of the sum of the absolute residuals. It is the multivariate generalisation of the median regression.
Value

A list including:

- `iters` The number of iterations that were required.
- `be` The beta coefficients.

Author(s)

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

References


See Also

`spat.med, sscov, lmfit`

Examples

```r
## Not run:
x <- as.matrix(iris[, 3:4])
y <- as.matrix(iris[, 1:2])
mod1 <- spatmed.reg(y, x)
## End(Not run)
```

Description

Spatial sign covariance matrix.

Usage

```r
sscov(x, me = NULL, tol = 1e-09)
```

Arguments

- `x` A matrix with continuous data.
- `me` If you have already computed the spatial median plug it in here.
- `tol` A tolerance level to terminate the process of finding the spatial median. This is set to 1e-09 by default.
Details

The spatial median is at first computed (if not supplied) and then the covariance matrix.

Value

The spatial sign covariance matrix.

Author(s)

Michail Tsagris

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

References


See Also

spat.med, spatmed.reg

Examples

res<-sscov( as.matrix(iris[, 1:4]) )

Description

It calculates, very fast, the (hyper-)spherical median of a sample.

Usage

mediandir(x)

Arguments

x The data, a numeric matrix with unit vectors.

Details

The "mediandir" employs a fixed point iterative algorithm stemming from the first derivative (Cabrera and Watson, 1990) to find the median direction as described in Fisher (1985) and Fisher, Lewis and Embleton (1987).
Value

The median direction.

Author(s)

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

References


See Also

vmf.mle

Examples

```r
m <- rnorm(3)
m <- m / sqrt( sum(m^2) )
x <- rvmf(100, m, 10)
res<-mediandir(x)
x <- NULL
```

Description

Standardisation.

Usage

```r
standardise(x, center = TRUE, scale = TRUE)
```

Arguments

- **x**: A matrix with data. It has to be matrix, if it is data.frame for example the function does not turn it into a matrix.
- **center**: Should the data be centred as well? TRUE or FALSE.
- **scale**: Should the columns have unit variance, yes (TRUE) or no (FALSE)?
Details

Similar to R’s built-in functions "scale" there is the option for centering or scaling only or both (default).

Value

A matrix with the standardised data.

Author(s)

Michail Tsagris

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

See Also

colVars, colmeans, colMads

Examples

```r
x <- matrnorm(100, 100)
a1 <- scale(x)[1:100, ]
a2 <- standardise(x)
all.equal(as.vector(a1), as.vector(a2))
x <- NULL
```

---

Sub-matrix

Description

Sub-matrix.

Usage

`submatrix(x, rowStart=1, rowEnd=1, colStart=1, colEnd=1)`

Arguments

- `x` A Matrix, List, Dataframe or Vector.
- `rowStart` Start of the row.
- `rowEnd` End of the row.
- `colStart` Start of the col.
- `colEnd` End of the col.
Sum of all pairwise distances in a distance matrix

Value

sub matrix like R’s, x[startrow:endrow,startcol:endcol]. Fast especially for big sub matrices.

Author(s)

Manos Papadakis
R implementation and documentation: Manos Papadakis <papadakm95@gmail.com>

See Also

Match,mvbetas,correls,univglms,colsums,colVars

Examples

x <- matrix( rnorm(100 * 100), ncol = 100 )
res<-submatrix(x,1,50,1,25) # x[1:50,1:25]
x<-NULL

Description

Sum of all pairwise distances in a distance matrix.

Usage

total.dist(x, method = "euclidean", square = FALSE, p = 0)
total.dista(x, y, square = FALSE)

Arguments

x A matrix with numbers.
y A second matrix with data. The number of columns of this matrix must be the same with the matrix x. The number of rows can be different.
method This is either "euclidean", "manhattan", "canberra1", "canberra2", "minimum", "maximum", "minkowski", "bhattacharyya", "hellinger", "total_variation" or "kullback_leibler/jensen_shannon". The last two options are basically the same.
square If you choose "euclidean" or "hellinger" as the method, then you can have the option to return the squared Euclidean distances by setting this argument to TRUE.
p This is for the the Minkowski, the power of the metric.
Details

In order to do the total.dist one would have to calculate the distance matrix and sum it. We do this internally in C++ without creating the matrix. For the total.dista it is the same thing.

Value

A numerical value, the sum of the distances.

Author(s)

Manos Papadakis

R implementation and documentation: Manos Papadakis <papadakm95@gmail.com>.

See Also

Dist, dista

Examples

```r
x <- matrix( rnorm(50 * 10), ncol = 10 )
res <- total.dist(x)
y <- matrix( rnorm(40 * 10), ncol = 10 )
res <- total.dista(x, y)
res <- total.dista(y, x)
```

```r
x <- y <- NULL
```

Description

Table Creation - Frequency of each value.

Usage

```r
Table(x, y=NULL, names = TRUE, useNA = FALSE, rm.zeros = FALSE)
Table.sign(x, names = TRUE, useNA = FALSE)
```

Arguments

- **x**: A vector with numeric/character data.
- **names**: A logical value (TRUE/FALSE) for add names.
- **y**: A vector with numeric/character data. Doesn’t work with “useNA”.
- **rm.zeros**: A logical value for removing zero columns/rows. Only for integer vectors for now.
Table Creation - Frequency of each value

useNA
Table: Integer/logical value:
FALSE: not NA values in vector. TRUE: count NAs and add the value in the last position of the returned vector. any other integer except 0,1: for just removing NAs.
Table.sign: Logical value, TRUE, for count NAs. Otherwise FALSE.
Doesn’t work character data.

Details
Like R’s "table":
for giving one argument,"x": If "names" is FALSE then, if "useNA" is TRUE then the NAs will be count, if is FALSE it means there are no NAs and for any other integer value the NAs will be ignored.
for giving two arguments,"x","y": If "names" is FALSE then, creates the contigency table, otherwise sets the col-row names with discrete values. If "rm.zeros" is FALSE then it won’t remove the zero columns/rows from the result but it will work only for positive integers for now. For this if "names" is TRUE then the col-row names will be the seq(min(),max()) for "x","y". In future updates it will be changed.
for both algorithms: You can’t use "useNA" with "names" for now. It is much faster to get the result without names (names = FALSE) but all the algorithms are more efficient than R’s.

Like R’s "table(sign())" but more efficient. Count the frequencies of positives, negatives, zeros and NAs values. If argument "names" is FALSE then the returned vector doesn’t have names. Otherwise ",-1,0,+1,NA". If "useNA" is TRUE then the NAs will be count, otherwise not. You can use "useNA" with "names".

Value
Table:
for giving one argument,"x": if "names" is TRUE then return a vector with names the discrete values of "x" and values there frequencies, otherwise only the frequencies
for giving two arguments,"x","y": if "names" is TRUE then return a contigency matrix with row-names the discrete values of "x", colnames the discrete values of "y" and values the frequencies of the pairs, otherwise only the frequencies of the pairs.
Table.sign: A vector with 4 values/frequencies: index 1: negatives index 2: zeros index 3: postives if "names" is TRUE then the returned vector have names ",-1,0,+1". if "useNA" is TRUE then 4th value has the frequencies of NAs and the returned vector will have one more name, ",-1,0,+1,NA", if "names" is also TRUE.

Author(s)
R implementation and documentation: Manos Papadakis <papadakm95@gmail.com>.

See Also
colShuffle,colVars,colmeans,read.directory,is_integer,as_integer
Tests for the dispersion parameter in Poisson distribution

Examples

```r
x<-runif(10)
y1<-Table(x)
y2<-as.vector(table(x)) # Needs a lot of time.
all.equal(y1,y2)
y1<-Table(x,names=FALSE)
all.equal(y1,y2) # the name attribute of y1 is null

y1<-Table.sign(x)
y2<-table(sign(x))
all.equal(y1,y2)

x<-y1<-y2<-NULL
```

Description

Tests for the dispersion parameter in Poisson distribution.

Usage

```r
poisdisp.test(y, alternative = "either", logged = FALSE)
pois.test(y, logged = FALSE)
```

Arguments

- `y`: A numerical vector with count data, 0, 1,...
- `alternative`: Do you want to test specifically for either over or underspersion ("either"), overdispersion ("over") or undersispersion ("under")?
- `logged`: Set to TRUE if you want the logarithm of the p-value.

Value

A vector with two elements, the test statistic and the (logged) p-value.

Author(s)

Michail Tsagris

R implementation and documentation: Michail Tsagris <mtsagris@uoc.gr> and Manos Papadakis <papadakm95@gmail.com>. 
Topological sort of a DAG

References


See Also

poisson.mle,negbin.mle,poisson.anova,poisson.anovas,poisson_only

Examples

```r
y <- rnbinom(500, 10, 0.6)
res<-poisdisp.test(y, "either")
res<-poisdisp.test(y, "over")
res<-pois.test(y)

y <- rpois(500, 10)
res<-poisdisp.test(y, "either")
res<-poisdisp.test(y, "over")
res<-pois.test(y)
```

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.
Transpose of a matrix

Description

Transpose of a matrix.

Usage

transpose(x)

Arguments

x  A numerical square matrix with data.

Value

The transposed matrix.
Uniformity test for circular data

Author(s)

Manos Papadakis
R implementation and documentation: Manos Papadakis <papadakm95@gmail.com>.

References


See Also

nth, colMaxs, colMins, colrange

Examples

x <- matrix(rnorm(500 * 500), ncol = 500, nrow=500)
system.time(transpose(x))
system.time(t(x))
x<-NULL

Uniformity test for circular data

Description

Hypothesis tests of uniformity for circular data.

Usage

kuiper(u)
watson(u)

Arguments

u A numeric vector containing the circular data which are expressed in radians.

Details

These tests are used to test the hypothesis that the data come from a circular uniform distribution.

Value

A vector with two elements, the value of the test statistic and its associated p-value.
Variance of a vector

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

References

See Also
vmf.mle, rvonmises

Examples
x <- rvonmises(n = 50, m = 2, k = 10)
res<-kuiper(x)
res<-watson(x)
x <- runif(50, 0, 2 * pi)
res<-kuiper(x)
res<-watson(x)

Description
Variance (and standard deviation) of a vector.

Usage
Var(x, std = FALSE, na.rm = FALSE)

Arguments
x
A vector with data.
std
If you want the standard deviation set this to TRUE, otherwise leave it FALSE.
na.rm
TRUE or FALSE for remove NAs if exists.

Details
This is a faster calculation of the usual variance of a matrix.

Value
The variance of the vector.
Vector allocation in a symmetric matrix

Author(s)

Michail Tsagris

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

See Also

colVars, cova

Examples

```r
x <- rnorm(100)
a1 <- Var(x)
a2 <- var(x)
x <- NULL
```

Vector allocation in a symmetric matrix

Description

Vector allocation in a symmetric matrix.

Usage

```r
squareform(x)
```

Arguments

x

An numerical vector whose size must be the one that matches the dimensions of the final matrix. See examples.

Details

The functions is written in C++ in order to be as fast as possible.

Value

A symmetric matrix. The vector is allocated in the upper and in the lower part of the matrix. The diagonal is filled with zeros.

Author(s)

R implementation and documentation: Manos Papadakis <papadakm95@gmail.com>.
Weibull regression model

See Also
colShuffle, colVars, colmeans

Examples

x <- rnorm(1)
res<-squareform(x) ## OK
x <- rnorm(3)
res<-squareform(x) ## OK
x <- rnorm(4)
res<-squareform(x) ## not OK

Description

Weibull regression model.

Usage

weib.reg(y, x, tol = 1e-07, maxiters = 100)

Arguments

y The dependent variable; a numerical vector with strictly positive data, i.e. greater than zero.

x A matrix with the data, where the rows denote the samples (and the two groups) and the columns are the variables. This can be a matrix or a data.frame (with factors).

tol The tolerance value to terminate the Newton-Raphson algorithm.

maxiters The max number of iterations that can take place in each regression.

Details

The function is written in C++ and this is why it is very fast. No standard errors are returned as they are not correctly estimated. We focused on speed.

Value

When full is FALSE a list including:

iters The iterations required by the Newton-Raphson.

loglik The log-likelihood of the model.

shape The shape parameter of the Weibull regression.

be The regression coefficients.
Yule’s Y (coefficient of colligation)

Author(s)
Stefanos Fafalios
R implementation and documentation: Stefanos Fafalios <stefanosfafalios@gmail.com>.

References

See Also
poisson_only, logistic_only, univglms, regression

Examples
```r
## Not run:
x <- matrix(rnorm(100 * 2), ncol = 2)
y <- rexp(100, 1)
res <- weib.reg(y, x)
## End(Not run)
```

Yule’s Y (coefficient of colligation)

Description
Yule’s Y (coefficient of colligation).

Usage
```r
yule(x)
```

Arguments
```r
x             A 2 x 2 matrix or a vector with 4 elements. In the case of the vector make sure it corresponds to the correct table.
```

Details
Yule’s coefficient of colligation is calculated.

Value
Yule’s Y is returned.
Yule’s Y (coefficient of colligation)

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

References

See Also
col.yule, odds.ratio

Examples
x <- rpois(4, 30) + 2
res<-yule(x)
res<-yule( matrix(x, ncol = 2) )
Index

!=.iterator (Iterator), 121
* 2 sample proportions tests
  Many 2 sample proportions tests, 136
* 2 variances test
  Many 2 sample tests, 137
* AR(1) model
  Estimation of an AR(1) model, 87
* All possible combinations
  All k possible combinations from n elements, 10
* Analysis of covariance
  Analysis of covariance, 11
  Many ANCOVAs, 140
* Analysis of variance
  Analysis of variance with a count variable, 12
* Angular central Gaussian distribution
  Angular central Gaussian random values simulation, 13
* Area under the curve
  Many (and one) area under the curve values, 135
* BIC
  BIC (using partial correlation) forward regression, 19
  BIC forward regression with generalised linear models, 20
* Beta distribution
  MLE of distributions defined in the (0, 1) interval, 198
* Beta function
  Natural logarithm of the beta function, 218
* Binary search Algorithm
  Binary search algorithm, 21
* Bradley-Terry model
  Fitted probabilities of the Terry-Bradley model, 97
* Canberra distance
  Distance matrix, 78
* Cauchy
  MLE of continuous univariate distributions defined on the real line, 195
* Checking Alias
  Check Namespace and Rd files, 26
* Checking Examples
  Check Namespace and Rd files, 26
* Checking Rd
  Check Namespace and Rd files, 26
* Checking R
  Check Namespace and Rd files, 26
* Checking Usage section
  Check Namespace and Rd files, 26
* Checking for FALSE
  Check Namespace and Rd files, 26
* Checking for TRUE
  Check Namespace and Rd files, 26
* Cholesky decomposition
  Cholesky decomposition of a square matrix, 31
* Circular data
  Column-wise uniformity Watson test for circular data, 58
  Uniformity test for circular data, 277
* Circular regression
  Circular or angular regression, 32
  Many simple circular or angular regressions, 164
* Circular-linear correlation
  Circular-linear correlation, 33
* Cochran’s Q test
  Many non parametric multi-sample tests, 152
* Column means
  Column and row-wise means of a
matrix, 37
* Column sums
  Column and row-wise sums of a matrix, 46
* Column-Row wise checking
  Check if any column or row is fill with values, 24
* Column-wise Any
  Column and row-wise Any/All, 36
* Column-wise Shuffle
  Column and row-wise Shuffle, 45
* Column-wise median absolute deviations
  Column and rows-wise mean absolute deviations, 49
* Column-wise medians
  Column and row-wise medians, 38
* Column-wise minimum
  Column-wise minimum and maximum, 54
* Column-wise nth
  Column and row-wise nth smallest value of a matrix/vector, 39
* Column-wise ranges
  Column and row-wise range of values of a matrix, 43
* Column-wise tabulate
  Column and row-wise tabulate, 47
* Column-wise true
  Column-wise true/false value, 57
* Column-wise variances
  Column and row-wise variances and standard deviations, 48
* Column-wise
  Column-wise MLE of some univariate distributions, 55
* Combinatorics
  All k possible combinations from n elements, 10
* Continuous distributions
  MLE of continuous univariate distributions defined on the positive line, 193
  MLE of continuous univariate distributions defined on the real line, 195
* Correlations
  Correlation between pairs of variables, 63
  Correlations, 65
* Covariance matrix
  Covariance and correlation matrix, 66
* Create - Fill
  Diagonal Matrix, 75
* DAG
  Topological sort of a DAG, 275
* Data Frame
  Index of the columns of a data.frame which are a specific type, 117
* Dataframe to Matrix
  Convert a dataframe to matrix, 60
* Deep copy
  Deep copy, 72
* Design Matrix
  Design Matrix, 74
* Determinant
  Check if any column or row is fill with values, 24
* Diagonal Matrix
  Diagonal Matrix, 75
* Differences
  Column-wise differences, 51
* Directional k-NN algorithm
  k-NN algorithm using the arc cosinus distance, 126
* Dirichlet distribution
  Fitting a Dirichlet distribution via Newton-Rapshon, 98
* Discrimination
  Prediction with some naive Bayes classifiers, 233
* Distance correlation
  Distance correlation, 77
* Distance covariance
  Distance variance and covariance, 80
* Distance matrix
  Distance matrix, 78
* Distance variance
  Distance variance and covariance, 80
* Distances
  Distance between vectors and a matrix, 76
  Sum of all pairwise distances in a distance matrix, 271
| *Divide and Qonquer* | Binary search algorithm, 21  
|                     | Find element, 95  
| *Eigenvalues* | Eigenvectors and eigenvectors in high dimensional principal component analysis, 81  
| *Energy distances* | Energy distance between matrices, 85  
| *Environment* | Deep copy, 72  
|                 | Iterator, 121  
|                 | Representation of Stack, 244  
| *Equality check* | Equality of objects, 86  
| *Euclidean distance* | Distance matrix, 78  
| *Exponential regressions* | Many exponential regressions, 142  
| *Export functions* | Insert/remove function names in/from the NAMESPACE file, 118  
|                 | Source many R files, 264  
| *Extract columns/rows* | Get specific columns/rows from a matrix, 107  
| *F-tests* | Many F-tests with really huge matrices, 143  
|                 | Many multi-sample tests, 150  
| *F-test* | Multi-sample tests for vectors, 209  
| *Factor variables* | Index of the columns of a data.frame which are a specific type, 117  
| *Factorials* | Binomial coefficient and its logarithm, 22  
| *Factor* | Fast and general – untyped representation of a factor variable, 93  
| *Find Value* | Find the given value in a hash table, 96  
| *Find element* | Find element, 95  
| *Floyd-Warshall algorithm* | Floyd-Warshall algorithm, 99  
| *Forward regression* | BIC (using partial correlation) forward regression, 19  
|                 | BIC forward regression with generalised linear models, 20  
|                 | Correlation based forward regression, 62  
|                 | Forward selection with generalised linear regression models, 101  
| *GLMS* | Many score based regressions, 161  
| *GLMs* | Quasi binomial regression for proportions, 234  
|                 | Quasi Poisson regression for count data, 236  
|                 | Significance testing for the coefficients of Quasi binomial or the quasi Poisson regression, 250  
| *G^2 test of conditional independence* | Chi-square and G-square tests of (unconditional) independence, 30  
|                 | G-square and Chi-square test of conditional independence, 102  
| *G^2 test of independence* | Matrix with G-square tests of independence, 183  
| *G^2 tests of independence* | Many G-square and Chi-square tests of independence, 144  
| *Gini coefficient* | Many Gini coefficients, 146  
| *Goodness of fit test* | Hypothesis test for von Mises–Fisher distribution over Kent distribution, 115  
| *Gumbel distribution* | MLE of continuous univariate distributions defined on the real line, 195  
| *Hash Function* | Find the given value in a hash table, 96  
|                 | Hash – Pair function, 108  


<table>
<thead>
<tr>
<th>Index Item</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hash tables</td>
<td>109</td>
</tr>
<tr>
<td>Hash object</td>
<td>109</td>
</tr>
<tr>
<td>Hash object to a list object</td>
<td>110</td>
</tr>
<tr>
<td>Hellinger distance</td>
<td>78</td>
</tr>
<tr>
<td>High dimensional data</td>
<td>111</td>
</tr>
<tr>
<td>Hypothesis testing</td>
<td>82</td>
</tr>
<tr>
<td>Empirical and exponential empirical likelihood tests for one sample</td>
<td>83</td>
</tr>
<tr>
<td>Empirical and exponential empirical likelihood tests for two samples</td>
<td>83</td>
</tr>
<tr>
<td>Many one sample tests</td>
<td>156</td>
</tr>
<tr>
<td>Hypothesis test</td>
<td>90</td>
</tr>
<tr>
<td>Exponential empirical likelihood for a one sample mean vector hypothesis testing</td>
<td>90</td>
</tr>
<tr>
<td>Integer variables</td>
<td>117</td>
</tr>
<tr>
<td>Index of the columns of a data.frame which are a specific type</td>
<td>117</td>
</tr>
<tr>
<td>Inverse matrix</td>
<td>120</td>
</tr>
<tr>
<td>Inverse of a symmetric positive definite matrix</td>
<td>120</td>
</tr>
<tr>
<td>Inverted Dirichlet distribution</td>
<td>201</td>
</tr>
<tr>
<td>MLE of the inverted Dirichlet distribution</td>
<td>201</td>
</tr>
<tr>
<td>James test</td>
<td>209</td>
</tr>
<tr>
<td>Multi-sample tests for vectors</td>
<td>209</td>
</tr>
<tr>
<td>Kent distribution</td>
<td>115</td>
</tr>
<tr>
<td>Hypothesis test for von Mises-Fisher distribution over Kent distribution</td>
<td>115</td>
</tr>
<tr>
<td>Laplace distribution</td>
<td>195</td>
</tr>
<tr>
<td>MLE of continuous univariate distributions defined on the real line</td>
<td>195</td>
</tr>
<tr>
<td>Linear mixed models</td>
<td>35</td>
</tr>
<tr>
<td>Column and row wise coefficients of variation</td>
<td>35</td>
</tr>
<tr>
<td>Many random intercepts LMMs for balanced data with a single identical covariate</td>
<td>157</td>
</tr>
<tr>
<td>Random intercepts linear mixed models</td>
<td>237</td>
</tr>
<tr>
<td>Logistic distribution</td>
<td>195</td>
</tr>
<tr>
<td>MLE of continuous univariate distributions defined on the real line</td>
<td>195</td>
</tr>
<tr>
<td>Logistic regressions</td>
<td>176</td>
</tr>
<tr>
<td>Many univariate simple logistic and Poisson regressions</td>
<td>176</td>
</tr>
<tr>
<td>Logistic regression</td>
<td>130</td>
</tr>
<tr>
<td>Logistic and Poisson regression models</td>
<td>130</td>
</tr>
<tr>
<td>Logistic or Poisson regression with a single categorical predictor</td>
<td>131</td>
</tr>
<tr>
<td>Lower and Upper triangular of a matrix</td>
<td>133</td>
</tr>
<tr>
<td>Lower and Upper triangular of a matrix</td>
<td>133</td>
</tr>
<tr>
<td>MCD estimation</td>
<td>111</td>
</tr>
<tr>
<td>High dimensional MCD based detection of outliers</td>
<td>111</td>
</tr>
<tr>
<td>Mahalanobis distance</td>
<td>134</td>
</tr>
<tr>
<td>Mahalanobis distance</td>
<td>134</td>
</tr>
<tr>
<td>Manhattan distance</td>
<td>78</td>
</tr>
<tr>
<td>Distance matrix</td>
<td>78</td>
</tr>
<tr>
<td>Many betas in regression</td>
<td>151</td>
</tr>
<tr>
<td>Many multivariate simple linear regressions coefficients</td>
<td>151</td>
</tr>
<tr>
<td>Many simple linear regressions coefficients</td>
<td>167</td>
</tr>
<tr>
<td>Match Function</td>
<td>180</td>
</tr>
<tr>
<td>Match</td>
<td>180</td>
</tr>
<tr>
<td>Matrices</td>
<td>237</td>
</tr>
<tr>
<td>Number of equal columns between</td>
<td>237</td>
</tr>
</tbody>
</table>
two matrices, 221
* McNemar’s test
   Many 2 sample tests, 137
* Median direction
   Spherical and hyperspherical median, 268
* Multinomial distribution
   MLE for multivariate discrete data, 190
     Multinomial regression, 211
* Multivariate analysis of variance
   James multivariate version of the t-test, 123
* Multivariate data
   Multivariate kurtosis, 212
* Multivariate hypothesis testing
   Exponential empirical likelihood hypothesis testing for two mean vectors, 91
* Multivariate normal distribution
   Density of the multivariate normal and t distributions, 73
   MLE of the multivariate (log-)normal distribution, 202
* Namespace file
   Check Namespace and Rd files, 26
     Insert/remove function names in/from the NAMESPACE file, 118
   Source many R files, 264
* Newton-Raphson
   Fitting a Dirichlet distribution via Newton-Raphson, 98
   MLE of distributions defined in the (0, 1) interval, 198
* Norm of a matrix
   Norm of a matrix, 220
* Numeric variables
   Index of the columns of a data.frame which are a specific type, 117
* Odds ratios
   Many odds ratio tests, 154
* Odds ratio
   Odds ratio and relative risk, 222
* One sample t-test
   One sample t-test for a vector, 223
* Orderings
   Column and row-wise Order - Sort
* Indices, 41
* Ordinal model
   MLE of the ordinal model without covariates, 205
* PC algorithm
   Skeleton of the PC algorithm, 256
* Pair Function
   Hash - Pair function, 108
* Pairs of vectors
   Column-row wise minima and maxima of two matrices, 50
   Minima and maxima of two vectors/matrices, 187
* Pareto
   MLE of continuous univariate distributions defined on the positive line, 193
* Pearson correlation
   Correlation based forward regression, 62
* Permutation Function
   Permutation, 228
* Poisson distribution
   Analysis of variance with a count variable, 12
     Many analysis of variance tests with a discrete variable, 139
     Many tests for the dispersion parameter in Poisson distribution, 171
   MLE of count data (univariate discrete distributions), 196
   Prediction with some naive Bayes classifiers, 233
   Tests for the dispersion parameter in Poisson distribution, 274
* Poisson regressions
   Many univariate simple quasi poisson regressions, 178
* Poisson regression
   Logistic or Poisson regression with a single categorical predictor, 131
* Poisson
   Forward selection with generalised linear regression models, 101
* Products
   Column and row-wise products, 42
* Quasi Poisson regression
  Quasi Poisson regression for count data, 236
* Quasi regression
  Quasi binomial regression for proportions, 234
  Significance testing for the coefficients of Quasi binomial or the quasi Poisson regression, 250
* Random values simulation
  Random values simulation from a von Mises distribution, 239
  Simulation of random values from a von Mises-Fisher distribution, 255
* Read Examples
  Reading the files of a directory, 241
* Read directory
  Reading the files of a directory, 241
* Remove functions
  Insert/remove function names in/from the NAMESPACE file, 118
* Repeated measures
  Many regression based tests for single sample repeated measures, 159
  Repeated measures anova, 242
* Replicate in columns/rows
  Replicate columns/rows, 243
* Round vector/matrix
  Round each element of a matrix/vector, 245
* Row - Wise matrix/vector count the frequency of a value
  Row - Wise matrix/vector count the frequency of a value, 246
* Row sums
  Column and row-wise sums of a matrix, 46
* Row-wise Any
  Column and row-wise Any/All, 36
* Row-wise Shuffle
  Column and row-wise Shuffle, 45
* Row-wise false
  Row-wise true value, 248
* Row-wise medians
  Column and row-wise medians, 38
* Row-wise minimum
  Row-wise minimum and maximum, 247
* Row-wise nth
  Column and row-wise nth smallest value of a matrix/vector, 39
* Row-wise tabulate
  Column and row-wise tabulate, 47
* Row-wise true-false
  Row-wise true value, 248
* Row-wise true
  Row-wise true value, 248
* Shapiro-Francia
  Many Shapiro-Francia normality tests, 163
* Significance testing
  Significance testing for the coefficients of Quasi binomial or the quasi Poisson regression, 250
* Simple linear regressions
  Many univariate simple linear regressions, 175
* Skewness coefficient
  Column-wise kurtosis and skewness coefficients, 52
* Skewness
  Hypothesis testing between two skewness or kurtosis coefficients, 116
  Skewness and kurtosis coefficients, 258
* Sort 2 vectors
  Sort - Integer Sort - Sort a vector corresponding to another, 260
* Sort function
  Sort and unique numbers, 262
* Sorting
  Sorting of the columns-rows of a matrix, 263
* Sort
  Sort - Integer Sort - Sort a vector corresponding to another, 260
* Stable Sort
  Sort - Integer Sort - Sort a
vector corresponding to another, 260
* Stack
  Representation of Stack, 244
* Standardisation
  Standardisation, 269
* Sub-matrix
  Sub-matrix, 270
* Sum
  Operations between two matrices or matrix and vector, 224
* Supervised classification
  k-NN algorithm using the arc cosine distance, 126
* Symmetric matrix
  Check whether a square matrix is symmetric, 29
* Table Creation
  Table Creation – Frequency of each value, 272
* Time series
  Estimation of an AR(1) model, 87
* Tobit model
  MLE of the tobit model, 206
* Topological sort
  Topological sort of a DAG, 275
* Transpose
  Transpose of a matrix, 276
* Two-way ANOVA
  Many two-way ANOVAs, 172
* Unequality of the covariance matrices
  James multivariate version of the t-test, 123
* Univariate normality test
  Many Shapiro-Francia normality tests, 163
* Variance components
  Moment and maximum likelihood estimation of variance components, 207
* Variance
  Some summary statistics of a vector for each level of a grouping variable, 259
  Variance of a vector, 278
* Weibull
  MLE of continuous univariate distributions defined on the positive line, 193
* Wigner semicircle distribution
  MLE of continuous univariate distributions defined on the real line, 195
* Zero range
  Search for variables with zero range in a matrix, 249
* analysis of variance
  Logistic or Poisson regression with a single categorical predictor, 131
  Many analysis of variance tests with a discrete variable, 139
  Many F-tests with really huge matrices, 143
  Many multi-sample tests, 150
  Many non-parametric multi-sample tests, 152
  Multi-sample tests for vectors, 209
* balanced design
  Column and row wise coefficients of variation, 35
  Many random intercepts LMMs for balanced data with a single identical covariate., 157
  Random intercepts linear mixed models, 237
* beta prime
  MLE of continuous univariate distributions defined on the positive line, 193
* bias corrected
  Distance correlation, 77
* binary data
  Forward selection with generalised linear regression models, 101
* binomial distribution
  MLE of count data (univariate discrete distributions), 196
* bivariate angular Gaussian
  MLE of some circular distributions, 200
* blocking ANOVA
  Many multi-sample tests, 150
  Multi-sample tests for vectors, 209
* categorical variables
  Many univariate simple linear
<table>
<thead>
<tr>
<th>Topic</th>
<th>Page(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>regressions</td>
<td>175</td>
</tr>
<tr>
<td>censored observations</td>
<td>MLE of the tobit model, 206</td>
</tr>
<tr>
<td>central angular Gaussian distribution</td>
<td>MLE of (hyper-)spherical distributions, 191</td>
</tr>
<tr>
<td>circular data</td>
<td>MLE of some circular distributions, 200</td>
</tr>
<tr>
<td>column-wise false</td>
<td>Column-wise true/false value, 57</td>
</tr>
<tr>
<td>column-wise maximum</td>
<td>Column-wise minimum and maximum, 54</td>
</tr>
<tr>
<td>column-wise minimum-maximum</td>
<td>Column-wise minimum and maximum, 54</td>
</tr>
<tr>
<td>column-wise true-false</td>
<td>Column-wise true/false value, 57</td>
</tr>
<tr>
<td>combinatorics</td>
<td>Binomial coefficient and its logarithm, 22</td>
</tr>
<tr>
<td>conditional MLE</td>
<td>Estimation of an AR(1) model, 87</td>
</tr>
<tr>
<td>continuous distributions</td>
<td>Column-wise MLE of some univariate distributions, 55</td>
</tr>
<tr>
<td>covariance matrix</td>
<td>Pooled covariance matrix, 232</td>
</tr>
<tr>
<td></td>
<td>Spatial sign covariance matrix, 267</td>
</tr>
<tr>
<td>cross-validation</td>
<td>Cross-Validation for the k-NN algorithm, 68</td>
</tr>
<tr>
<td>data check</td>
<td>Search for variables with zero range in a matrix, 249</td>
</tr>
<tr>
<td>density values</td>
<td>Density of the multivariate normal and t distributions, 73</td>
</tr>
<tr>
<td>dependent binary data</td>
<td>Multi-sample tests for vectors, 209</td>
</tr>
<tr>
<td>derivatives</td>
<td>Natural logarithm of the gamma function and its derivatives, 219</td>
</tr>
<tr>
<td>digamma function</td>
<td>Natural logarithm of the gamma function and its derivatives, 219</td>
</tr>
<tr>
<td>directed graph</td>
<td>Floyd-Warshall algorithm, 99</td>
</tr>
<tr>
<td>directional data</td>
<td>Angular central Gaussian random values simulation, 13</td>
</tr>
<tr>
<td></td>
<td>MLE of (hyper-)spherical distributions, 191</td>
</tr>
<tr>
<td>discrete distributions</td>
<td>Column-wise MLE of some univariate distributions, 55</td>
</tr>
<tr>
<td>dispersion parameter</td>
<td>Many tests for the dispersion parameter in Poisson distribution, 171</td>
</tr>
<tr>
<td></td>
<td>Tests for the dispersion parameter in Poisson distribution, 274</td>
</tr>
<tr>
<td>equality of variances</td>
<td>Many multi-sample tests, 150</td>
</tr>
<tr>
<td></td>
<td>Multi-sample tests for vectors, 209</td>
</tr>
<tr>
<td>excessive zeros</td>
<td>MLE of count data (univariate discrete distributions), 196</td>
</tr>
<tr>
<td>fitted probabilities</td>
<td>Fitted probabilities of the Terry-Bradley model, 97</td>
</tr>
<tr>
<td>folded normal</td>
<td>MLE of continuous univariate distributions defined on the positive line, 193</td>
</tr>
<tr>
<td>fractional response</td>
<td>Quasi binomial regression for proportions, 234</td>
</tr>
<tr>
<td></td>
<td>Significance testing for the coefficients of Quasi binomial or the quasi Poisson regression, 250</td>
</tr>
<tr>
<td>gamma distribution</td>
<td>MLE of continuous univariate distributions defined on the positive line, 193</td>
</tr>
<tr>
<td>generalised linear models</td>
<td>Logistic and Poisson regression models, 130</td>
</tr>
<tr>
<td></td>
<td>Many univariate simple logistic and Poisson regressions, 176</td>
</tr>
<tr>
<td></td>
<td>Many univariate simple quasi poisson regressions, 178</td>
</tr>
<tr>
<td>geometric distribution</td>
<td>Analysis of variance with a count</td>
</tr>
</tbody>
</table>
INDEX

variable, 12
Many analysis of variance tests with a discrete variable, 139
MLE of count data (univariate discrete distributions), 196

* groupings
Some summary statistics of a vector for each level of a grouping variable, 259

* half normal
MLE of continuous univariate distributions defined on the positive line, 193

* harmonic means
Column and row-wise means of a matrix, 37

* high dimensional data
Eigenvalues and eigenvectors in high dimensional principal component analysis, 81

* huge datasets
Many F-tests with really huge matrices, 143

* hypersecant distribution for proportions
MLE of distributions defined in the (0, 1) interval, 198

* hypothesis testing
Column-wise uniformity Watson test for circular data, 58
Hypothesis testing between two skewness or kurtosis coefficients, 116
Uniformity test for circular data, 277

* inflated beta distribution
MLE of distributions defined in the (0, 1) interval, 198

* interaction
Many two-way ANOVAs, 172

* is_integer
Check if values are integers and convert to integer, 25

* iterator
Iterator, 121

* k-NN algorithm
Cross-Validation for the k-NN algorithm, 68
k nearest neighbours algorithm (k-NN), 124

* kurtosis coefficient
Column-wise kurtosis and skewness coefficients, 52

* kurtosis
Hypothesis testing between two skewness or kurtosis coefficients, 116
Multivariate kurtosis, 212
Skewness and kurtosis coefficients, 258

* large scale data
Linear models for large scale data, 128

* left censoring
MLE of the tobit model, 206

* list
Hash object, 109
Hash object to a list object, 110

* logarithm
Natural logarithm of the beta function, 218

* logistic normal distribution
MLE of distributions defined in the (0, 1) interval, 198

* matrix
Column and row-wise Order - Sort Indices, 41
Column and row-wise products, 42
Column-wise differences, 51
Transpose of a matrix, 276

* maximum frequency
Minimum and maximum frequencies, 189

* maximum likelihood estimation
Column and row wise coefficients of variation, 35
Fitting a Dirichlet distribution via Newton-Rapshon, 98
Many random intercepts LMMs for balanced data with a single identical covariate, 157
MLE of (hyper-)spherical distributions, 191
MLE of distributions defined in the (0, 1) interval, 198
Moment and maximum likelihood estimation of variance, 313
components, 207
Random intercepts linear mixed models, 237

* maximum
Column-row wise minima and maxima of two matrices, 50
Minima and maxima of two vectors/matrices, 187
Minimum and maximum, 188

* mean vector
Exponential empirical likelihood for a one sample mean vector hypothesis testing, 90

* minimum frequency
Minimum and maximum frequencies, 189

* minimum or maximum of negative
Apply method to Positive and Negative number, 15

* minimum or maximum of positive
Apply method to Positive and Negative number, 15

* minimum
Column-row wise minima and maxima of two matrices, 50
Minima and maxima of two vectors/matrices, 187
Minimum and maximum, 188

* moments estimation
Moment and maximum likelihood estimation of variance components, 207

* multinominal distribution
Prediction with some naive Bayes classifiers, 233

* multinominal regressions
Many score based regressions, 161

* multivariate Laplace distribution
Multivariate Laplace random values simulation, 213

* multivariate discrete data
MLE for multivariate discrete data, 190

* multivariate normal distribution
Multivariate normal and t random values simulation, 214

* multivariate t distribution
Density of the multivariate normal and t distributions, 73

* naive Bayes
Prediction with some naive Bayes classifiers, 233

* negative binomial
MLE of count data (univariate discrete distributions), 196

* negative numbers
Apply method to Positive and Negative number, 15

* non parametric statistics
Many non parametric multi-sample tests, 152

* non parametric test
Empirical and exponential empirical likelihood tests for one sample, 82

Empirical and exponential empirical likelihood tests for two samples, 83

Exponential empirical likelihood hypothesis testing for two mean vectors, 91

* normal distribution
Prediction with some naive Bayes classifiers, 233

* nth elements
Column and row-wise nth smallest value of a matrix/vector, 39

Median of a vector, 186

* one sample
Empirical and exponential empirical likelihood tests for one sample, 82

Many one sample tests, 156

* operations
Operations between two matrices or matrix and vector, 224

* outliers
High dimensional MCD based detection of outliers, 111

* partial correlation
BIC (using partial correlation) forward regression, 19
Correlation based forward regression, 62

* percentages
Hypothesis test for two means of
percentages. 114
Many hypothesis tests for two means of percentages, 147

* poisson regression
  Logistic and Poisson regression models, 130

* positive definite
  Inverse of a symmetric positive definite matrix, 120

* positive multivariate data
  MLE of the inverted Dirichlet distribution, 201

* positive numbers
  Apply method to Positive and Negative number, 15

* projected normal distribution
  MLE of (hyper-)spherical distributions, 191

* projected normal
  Circular or angular regression, 32
  Many simple circular or angular regressions, 164

* proportion test
  Many one sample tests, 156

* proportional odds
  MLE of the ordinal model without covariates, 205

* proportions
  Forward selection with generalised linear regression models, 101
  MLE of distributions defined in the (0, 1) interval, 198

* random values simulation
  Angular central Gaussian random values simulation, 13
  Multivariate Laplace random values simulation, 213
  Multivariate normal and t random values simulation, 214

* regression
  Many regression based tests for single sample repeated measures, 159
  Multinominal regression, 211
  Repeated measures anova, 242

* robust statistics
  Pooled covariance matrix, 232
  Spatial median for Euclidean data, 265
  Spatial sign covariance matrix, 267

* row means
  Column and row-wise means of a matrix, 37

* row-wise maximum
  Row-wise minimum and maximum, 247

* row-wise variances
  Column and row-wise variances and standard deviations, 48

* score based tests
  Many score based regressions, 161

* shortest paths
  Floyd-Warshall algorithm, 99

* single categorical predictor
  Logistic or Poisson regression with a single categorical predictor, 131

* sorting
  Median of a vector, 186

* spatial median
  Spatial median for Euclidean data, 265

* spherical data
  MLE of (hyper-)spherical distributions, 191

* summary statistics
  Many regression based tests for single sample repeated measures, 159
  Repeated measures anova, 242

* symmetric matrix
  Inverse of a symmetric positive definite matrix, 120
  Vector allocation in a symmetric matrix, 279

* t distribution
  MLE of continuous univariate distributions defined on the real line, 195

* t-tests
  Many 2 sample tests, 137
  Many hypothesis tests for two means of percentages, 147
  Matrix with all pairs of t-tests, 182

* t-test
  Hypothesis test for two means of...
percentages, 114
Many one sample tests, 156
* total sum
   Energy distance between matrices, 85
   Sum of all pairwise distances in a distance matrix, 271
* trigamma function
   Natural logarithm of the gamma function and its derivatives, 219
* two samples
   Empirical and exponential empirical likelihood tests for two samples, 83
* uniformity tests
   Column-wise uniformity Watson test for circular data, 58
* uniformity test
   Uniformity test for circular data, 277
* unique numbers
   Sort and unique numbers, 262
* univariate approach
   Many regression based tests for single sample repeated measures, 159
   Repeated measures anova, 242
* variable selection
   Forward selection with generalised linear regression models, 101
* variance test
   Many one sample tests, 156
* variances of many samples
   Column and row-wise variances and standard deviations, 48
* von Mises distribution
   MLE of some circular distributions, 200
* von Mises-Fisher distribution
   Hypothesis test for von Mises-Fisher distribution over Kent distribution, 115
   MLE of (hyper-)spherical distributions, 191
   Random values simulation from a von Mises distribution, 239
   Simulation of random values from a von Mises-Fisher distribution, 255
* wrapped Cauchy distribution
   MLE of some circular distributions, 200
* zero inflated Poisson
   MLE of count data (univariate discrete distributions), 196
* zero truncated Poisson
   MLE of count data (univariate discrete distributions), 196
.lm.fit, 129
==.iterator (Iterator), 121
[.Hash (Hash object), 109
[.ufactor (Fast and general - untyped representation of a factor variable), 93
[<-.Hash (Hash object), 109

AddToNamespace (Insert/remove function names in/from the NAMESPACE file), 118
env.copy (Deep copy), 72
RemoveFromNamespace (Insert/remove function names in/from the NAMESPACE file), 118
Stack (Representation of Stack), 244

acg.mle, 14, 33, 165
acg.mle (MLE of (hyper-)spherical distributions), 191
AddToNamespace, 28, 242, 265
All k possible combinations from n elements, 10
all_equals (Equality of objects), 86
allbetas, 64, 65, 106, 129, 152, 167, 170, 174–177, 179, 180
allbetas (Many simple linear regressions coefficients), 167
alltttests (Matrix with all pairs of t-tests), 182
Analysis of covariance, 11
Analysis of variance with a count variable, 12
ancova1 (Analysis of covariance), 11
ancovas, 12, 173
ancovas (Many ANCOVAs), 140
Angular central Gaussian random values simulation, 13
anova, 13, 98, 132, 139
ANOVA for two quasi Poisson regression models, 14
anoval, 12, 224
anoval (Multi-sample tests for vectors), 209
anova_propreg, 235
anova_propreg (Significance testing for the coefficients of Quasi binomial or the quasi Poisson regression), 250
anova_qposis.reg (ANOVA for two quasi Poisson regression models), 14
anovas, 138, 141
anovas (Many multi-sample tests), 150
Apply method to Positive and Negative number, 15
Apply to each column a method under condition, 17
apply.condition (Apply to each column a method under condition), 17
ar1 (Estimation of an AR(1) model), 87
as.Rfast.function (Convert R function to the Rfast’s corresponding), 61
as_integer, 26, 273
as_integer (Check if values are integers and convert to integer), 25
auc, 89
auc (Many (and one) area under the curve values), 135
Backward selection regression, 18
bc (Estimation of the Box-Cox transformation), 88
bcdcor, 113
bcdcor (Distance correlation), 77
beta.mle, 99, 194, 205, 218, 219
beta.mle (MLE of distributions defined in the (0, 1) interval), 198
betabinom.mle (MLE of count data (univariate discrete distributions)), 196
betageom.mle (MLE of count data (univariate discrete distributions)), 196
betapriceme (MLE of continuous univariate distributions defined on the positive line), 193
BIC (using partial correlation) forward regression, 19
BIC forward regression with generalised linear models, 20
bic.corfsreg, 21
bic.corfsreg (BIC (using partial correlation) forward regression), 19
bic.fs.reg (BIC forward regression with generalised linear models), 20
Binary search algorithm, 21
binary_search, 96, 247
binary_search (Binary search algorithm), 21
bincomb (Permutation), 228
binom.mle (MLE of count data (univariate discrete distributions)), 196
Binomial coefficient and its logarithm, 22
block.anova (Multi-sample tests for vectors), 209
block.anovas, 153
block.anovas (Many multi-sample tests), 150
boot.ttest2, 90, 142
boot.ttest2 (Bootstrap t-test for 2 independent samples), 23
Bootstrap t-test for 2 independent samples, 23
borel.mle (MLE of count data (univariate discrete distributions)), 196
bs.reg (Backward selection regression), 18
btmprobs (Fitted probabilities of the Terry-Bradley model), 97
cat.goftests (Many one sample goodness of fit tests for categorical data), 155
cauchy.mle (MLE of continuous univariate distributions defined on the real line), 195
Check if any column or row is fill with values, 24
Check if values are integers and convert to integer, 25
Check Namespace and Rd files, 26
Check whether a square matrix is symmetric, 29
check_data (Search for variables with zero range in a matrix), 249
checkAliases (Check Namespace and Rd files), 26
checkExamples (Check Namespace and Rd files), 26
checkExamples, 242
checkNamespace (Check Namespace and Rd files), 26
checkRd, 242
checkTF (Check Namespace and Rd files), 26
checkUsage (Check Namespace and Rd files), 26
Chi-square and G-square tests of (unconditional) indepdence, 30
chi2Test (G-square and Chi-square test of conditional indepdence), 102
chi2Test_univariate (Matrix with G-square tests of indepdence), 183
chi2tests (Many G-square and Chi-square tests of indepdence), 144
chisq.mle (MLE of continuous univariate distributions defined on the positive line), 193
cholesky, 29, 121
cholesky (Cholesky decomposition of a square matrix), 31
Cholesky decomposition of a square matrix, 31
Choose, 217, 246
Choose (Binomial coefficient and its logarithm), 22
circlin.cor (Circular-linear correlation), 33
Circular or angular regression, 32
Circular-linear correlation, 33

col.coxpoisrat (Cox confidence interval for the ratio of two Poisson variables), 67
col.yule, 282
col.yule (Column-wise Yule's Y (coefficient of colligation)), 59
colAll (Column and row-wise Any/All), 36
colanovas (Many Welch's F-tests), 179
colAny (Column and row-wise Any/All), 36
colar1 (Estimation of an AR(1) model), 87
colaucs (Many (and one) area under the curve values), 135
colCountValues (Row - Wise matrix/vector count the frequency of a value), 246
colCumMaxs (Column-wise cumulative operations (sum, prod, min, max)), 34
colCumMins (Column-wise cumulative operations (sum, prod, min, max)), 34
colCumProds (Column-wise cumulative operations (sum, prod, min, max)), 34
colCumSums (Column-wise cumulative operations (sum, prod, min, max)), 34
colcvss (Column and row wise coefficients of variation), 35
coldiffs, 41, 43
coldiffs (Column-wise differences), 51
colexp2.mle (Column-wise MLE of some univariate distributions), 55
colexpml (Column-wise MLE of some univariate distributions), 55
colFalse, 76, 133, 249
colFalse (Column-wise true/false value), 57
colgammamle (Column-wise MLE of some univariate distributions), 55
colgeom.mle (MLE for multivariate discrete data), 190
colhameans (Column and row-wise means of a matrix), 37
colinvgauss.mle (Column-wise MLE of some univariate...
INDEX

distributions, 55
colkurtosis (Column-wise kurtosis and skewness coefficients), 52
collaplace.mle (Column-wise MLE of some univariate distributions), 55
collindley.mle (Column-wise MLE of some univariate distributions), 55
colMads, 38, 185, 225, 270
colMads (Column and rows-wise mean absolute deviations), 49
colmaxboltz.mle (Column-wise MLE of some univariate distributions), 55
colMaxs, 10, 43, 50, 248, 264, 277
colMaxs (Column-wise minimum and maximum), 54
colMeans, 37, 39, 40, 46, 49
colmeans, 26, 46, 47, 49, 51, 52, 72, 112, 116, 118, 122, 135, 146, 212, 216, 220, 221, 225, 245, 259, 260, 270, 273, 280
colmeans (Column and row-wise means of a matrix), 37
colMedians, 17, 25, 35, 37, 38, 40, 41, 43, 46, 49, 50, 52, 55, 57, 61, 79, 108, 112, 116, 185, 186, 188, 189, 212, 221, 244, 248, 259, 260, 266
colMedians (Column and row-wise medians), 38
colMins, 10, 38, 43, 50, 187, 248, 264, 277
colMins (Column-wise minimum and maximum), 54
colMinsMaxs (Column-wise minimum and maximum), 54
colnormal.mle (Column-wise MLE of some univariate distributions), 55
colnormlog.mle (Column-wise MLE of some univariate distributions), 55
colnth, 16, 261
colnth (Column and row-wise nth smallest value of a matrix/vector), 39
colOrder (Column and row-wise Order - Sort Indices), 41
colpareto.mle (Column-wise MLE of some univariate distributions), 55
colPmax (Column-row wise minima and maxima of two matrices), 50
colPmin (Column-row wise minima and maxima of two matrices), 50
colpois.tests (Many tests for the dispersion parameter in Poisson distribution), 171
colpoisdisp.tests (Many tests for the dispersion parameter in Poisson distribution), 171
colpoisson.anovas (Many ANOVAS for count data with Poisson or quasi Poisson models), 141
colpoisson.mle, 233
colpoisson.mle (MLE for multivariate discrete data), 190
colprods, 41
colprods (Column and row-wise products), 42
colquasipoisson.anovas (Many ANOVAS for count data with Poisson or quasi Poisson models), 141
colrange, 10, 25, 49, 55, 57, 108, 188, 189, 198, 244, 250, 264, 277
colrange (Column and row-wise range of values of a matrix), 43
colRanks, 241
colRanks (Column and row-wise ranks), 44
colrayleigh.mle (Column-wise MLE of some univariate distributions), 55
colrint.regbx, 149
colrint.regbx (Many random intercepts LMMs for balanced data with a single identical covariate.), 157
colrow.value (Check if any column or row is fill with values), 24
colShuffle, 47, 72, 118, 122, 245, 273, 280
colShuffle (Column and row-wise Shuffle), 45
colskewness, 116, 146, 212, 259
colskewness (Column-wise kurtosis and skewness coefficients), 52
colSort, 25, 43, 55, 57, 76, 100, 108, 187–189, 244, 248, 263
colSort (Sorting of the columns-rows of a matrix), 263
colsums, 17, 35, 36, 38, 41, 43, 61, 87, 168, 181, 271
colsums (Column and row-wise sums of a matrix), 46
colTabulate, 54
colTabulate (Column and row-wise tabulate), 47
colTrue, 76, 133, 249
colTrue (Column-wise true/false value), 57
colTrueFalse (Column-wise true/false value), 57
Column-wise cumulative operations (sum, prod, min, max), 34
Column and row wise coefficients of variation, 35
Column and row-wise Any/All, 36
Column and row-wise means of a matrix, 37
Column and row-wise medians, 38
Column and row-wise nth smallest value of a matrix/vector, 39
Column and row-wise Order - Sort Indices, 41
Column and row-wise products, 42
Column and row-wise range of values of a matrix, 43
Column and row-wise ranks, 44
Column and row-wise Shuffle, 45
Column and row-wise sums of a matrix, 46
Column and row-wise tabulate, 47
Column and row-wise variances and standard deviations, 48
Column and rows-wise mean absolute deviations, 49
Column-row wise minima and maxima of two matrices, 50
Column-wise differences, 51
Column-wise kurtosis and skewness coefficients, 52
Column-wise matching coefficients, 53
Column-wise minimum and maximum, 54
Column-wise MLE of some univariate distributions, 55
Column-wise true/false value, 57
Column-wise uniformity Watson test for circular data, 58
Column-wise Yule’s Y (coefficient of colligation), 59
columns (Get specific columns/rows for a matrix), 107
colvarcomps.mle, 158, 208
colvarcomps.mle (Many moment and maximum likelihood estimations of variance components), 148
colvarcomps.mom, 239
colvarcomps.mom (Many moment and maximum likelihood estimations of variance components), 148
colVars (Column and row-wise variances and standard deviations), 48
colvm.mle (Column-wise MLE of some univariate distributions), 55
colwatsions (Column-wise uniformity Watson test for circular data), 58
colweibull.mle (Column-wise MLE of some univariate distributions), 55
comb_n, 23, 229
comb_n (All k possible combinations from n elements), 10
combn, 229
Convert a dataframe to matrix, 60
Convert R function to the Rfast’s corresponding, 61
cor, 66, 67
cor.fbed, 227
cor.fbed (FBED variable selection method using the correlation), 94
cor.fsreg, 18, 20, 21, 95, 102, 126, 129, 227
cor.fsreg (Correlation based forward regression), 62
cora, 29, 238
cora (Covariance and correlation matrix), 66
corpairs, 146
corpairs (Correlation between pairs of variables), 63
Correlation based forward regression, 62
Correlation between pairs of
variables, 63
Correlations, 65
correls, 45, 64, 68, 87, 89, 95, 103, 129, 146, 152, 168, 174, 176, 177, 180, 184, 227, 231, 241, 258, 271
correls (Correlations), 65
count_value (Row - Wise matrix/vector count the frequency of a value), 246
cov, 67
cova, 29, 121, 279
cova (Covariance and correlation matrix), 66
Covariance and correlation matrix, 66
Cox confidence interval for the ratio of two Poisson variables, 67
cox.poisrat (Cox confidence interval for the ratio of two Poisson variables), 67
cqtest (Multi-sample tests for vectors), 209
cqtests (Many non parametric multi-sample tests), 152
Cross-Validation for the k-NN algorithm, 68
Cross-Validation for the k-NN algorithm using the arc cosinus distance, 70
Crossprod, 66, 81
Crossprod (Matrix multiplication), 181
cct.mle (MLE of continuous univariate distributions defined on the real line), 195
data.frame.to_matrix, 256
data.frame.to_matrix (Convert a dataframe to matrix), 60
dcor, 80
dcor (Distance correlation), 77
dcor.ttest, 78
dcor.ttest (Hypothesis test for the distance correlation), 112
dcov, 78, 113
dcov (Distance variance and covariance), 80
Deep copy, 72
Density of the multivariate normal and t distributions, 73
Design Matrix, 74
design_matrix (Design Matrix), 74
Diag.fill, 225
Diag.fill (Diagonal Matrix), 75
Diag.matrix (Diagonal Matrix), 75
Diagonal Matrix, 75
Digamma (Natural logarithm of the gamma function and its derivatives), 219
diri.nr2, 199, 202, 205, 218, 219
diri.nr2 (Fitting a Dirichlet distribution via Newton-Rapshon), 98
dirimultinom.mle (MLE for multivariate discrete data), 190
dirknn, 72, 126
dirknn (k-NN algorithm using the arc cosinus distance), 126
dirknn.cv, 70, 127
dirknn.cv (Cross-Validation for the k-NN algorithm using the arc cosinus distance), 70
Dist, 51, 70, 77, 86, 220, 225, 272
Dist (Distance matrix), 78
dista, 51, 70, 79, 86, 135, 220, 225, 272
dista (Distance between vectors and a matrix), 76
Distance between vectors and a matrix, 76
Distance correlation, 77
Distance matrix, 78
Distance variance and covariance, 80
dmvnorm, 203, 204
dmvnorm (Density of the multivariate normal and t distributions), 73
dmvtnorm (Density of the multivariate normal and t distributions), 73
dvar, 86
dvar (Distance variance and covariance), 80
eachcol.apply (Operations between two matrices or matrix and vector), 224
eachrow (Operations between two matrices or matrix and vector), 224
edist, 78, 80, 113
edist (Energy distance between matrices), 85
eel.test1 (Empirical and exponential empirical likelihood tests for one sample), 82
eel.test2 (Empirical and exponential empirical likelihood tests for two samples), 83
eigen.sym (Limited number of eigenvalues and eigenvectors of a symmetric matrix), 127
Eigenvalues and eigenvectors in high dimensional principal component analysis, 81
el.test1 (Empirical and exponential empirical likelihood tests for one sample), 82
el.test2 (Empirical and exponential empirical likelihood tests for two samples), 83
Elem (Iterator), 121
Elem<- (Iterator), 121
Empirical and exponential empirical likelihood tests for one sample, 82
Empirical and exponential empirical likelihood tests for two samples, 83
Energy distance between matrices, 85
Equality of objects, 86
Estimation of an AR(1) model, 87
Estimation of the Box-Cox transformation, 88
Exact t-test for 2 independent samples, 89
exact.ttest2, 24
exact.ttest2 (Exact t-test for 2 independent samples), 89
exp2.mle (MLE of continuous univariate distributions defined on the positive line), 193
expmle (MLE of continuous univariate distributions defined on the positive line), 193
Exponential empirical likelihood for a one sample mean vector hypothesis testing, 90
Exponential empirical likelihood hypothesis testing for two mean vectors, 91
expregs (Many exponential regressions), 142
factor, 94
Fast and general - untyped representation of a factor variable, 93
FBED variable selection method using the correlation, 94
Find element, 95
Find the given value in a hash table, 96
fish.kent (Hypothesis test for von Mises-Fisher distribution over Kent distribution), 115
Fitted probabilities of the Terry-Bradley model, 97
Fitting a Dirichlet distribution via Newton-Rapshon, 98
floyd, 276
floyd (Floyd-Warshall algorithm), 99
Floyd-Warshall algorithm, 99
foldnorm.mle (MLE of continuous univariate distributions defined on the positive line), 193
Forward selection with generalised linear regression models, 101
freq.max (Minimum and maximum frequencies), 189
freq.min (Minimum and maximum frequencies), 189
fs.reg, 18, 21, 95, 126, 227
fs.reg (Forward selection with generalised linear regression models), 101
ftest, 24, 83, 90, 114, 142, 148
ftest (Multi-sample tests for vectors), 209
ftests, 12, 84, 136–138, 141, 143, 144, 153, 155, 157, 164, 173, 179, 180, 183, 210
ftests (Many multi-sample tests), 150
G-square and Chi-square test of conditional independence, 102
g2Test, 12, 13, 30, 146, 184, 223, 256, 258
g2Test (G-square and Chi-square test of conditional independence), 102
g2Test_perm, 146, 184

g2Test_perm (G-square and Chi-square test of conditional independence), 102

g2Test_univariate, 30, 102, 103, 154, 183, 258

g2Test_univariate (Matrix with G-square tests of independence), 183

g2Test_univariate_perm, 30, 102, 103

g2Test_univariate_perm (Matrix with G-square tests of independence), 183

g2tests, 98, 139

g2tests (Many G-square and Chi-square tests of independence), 144

g2tests_perm (Many G-square and Chi-square tests of independence), 144

Gamma regression with a log-link, 104

gammacon (Gamma regression with a log-link), 104

gammamle, 56, 196, 207

gammamle (MLE of continuous univariate distributions defined on the positive line), 193

gammanb (Naive Bayes classifiers), 215

gammanb.pred (Prediction with some naive Bayes classifiers), 233

gammareg (Gamma regression with a log-link), 104

gammaregs, 105

gammaregs (Many simple regressions for positive valued data), 169

Gaussian regression with a log-link, 105

gaussian.nb, 203, 204, 233

gaussian.nb (Naive Bayes classifiers), 215

gaussiannb.pred, 216

gaussiannb.pred (Prediction with some naive Bayes classifiers), 233

gchi2Test (Chi-square and G-square tests of (unconditional) independence), 30

Generates random values from a normal and puts them in a matrix, 106

geom.anova (Analysis of variance with a count variable), 12

geom.anovas (Many analysis of variance tests with a discrete variable), 139

geom.mle (MLE of count data (univariate discrete distributions)), 196

geom.nb (Naive Bayes classifiers), 215

geom.regs (Many simple geometric regressions), 165

gemnb.pred (Prediction with some naive Bayes classifiers), 233

Get specific columns/rows for a matrix, 107

ginis (Many Gini coefficients), 146

glm_logistic, 102, 211

glm_logistic (Logistic and Poisson regression models), 130

glm_poisson, 102

glm_poisson (Logistic and Poisson regression models), 130

group (Some summary statistics of a vector for each level of a grouping variable), 259

groupcorrels (Correlations), 65

gumbel.mle (MLE of continuous univariate distributions defined on the real line), 195

halfnorm.mle (MLE of continuous univariate distributions defined on the positive line), 193

Hash (Hash object), 109

Hash - Pair function, 108

Hash object, 109

Hash object to a list object, 110

hash.find, 109–111

hash.find (Find the given value in a hash table), 96

hash.list, 97, 110, 111

hash.list (Hash - Pair function), 108

hash2list (Hash object to a list object), 110

hd.eigen, 128

hd.eigen (Eigenvalues and eigenvectors in high dimensional principal component analysis), 81

High dimensional MCD based detection of outliers, 111
hsecant01.mle (MLE of distributions defined in the (0, 1) interval), 198
Hypothesis test for the distance correlation, 112
Hypothesis test for two means of percentages, 114
Hypothesis test for von Mises-Fisher distribution over Kent distribution, 115
Hypothesis testing between two skewness or kurtosis coefficients, 116
iag.mle, 33, 74, 115, 165, 256
iag.mle (MLE of (hyper-)spherical distributions), 191
ibeta.mle (MLE of distributions defined in the (0, 1) interval), 198
Index of the columns of a data.frame which are a specific type, 117
Insert/remove function names in/from the NAMESPACE file, 118
invdir.mle (MLE of the inverted Dirichlet distribution), 201
Inverse Gaussian regression with a log-link, 119
Inverse of a symmetric positive definite matrix, 120
invgauss.mle (MLE of continuous univariate distributions defined on the positive line), 193
invgauss.reg, 105
invgauss.reg (Inverse Gaussian regression with a log-link), 119
invgauss.regs, 120
invgauss.regs (Many simple regressions for positive valued data), 169
is.symmetric, 31, 60
is.symmetric (Check whether a square matrix is symmetric), 29
is_element, 22
is_element (Find element), 95
is_integer, 273
is_integer (Check if values are integers and convert to integer), 25
Lambda, 23
Lbeta, 23, 217
Lbinomial (Beta function), 218
Lchoose, 217, 246
Lchoose (Binomial coefficient and its logarithm), 22
length.Hash (Hash object), 109
Lgamma, 23, 218
Lgamma (Natural logarithm of the gamma function and its derivatives), 219
Limited number of eigenvalues and eigenvectors of a symmetric matrix, 127
Iterate, 121
iterator (Iterator), 121
James, 91, 93
James (James multivariate version of the t-test), 123
James multivariate version of the t-test, 123
k nearest neighbours algorithm (k-NN), 124
k-NN algorithm using the arc cosine distance, 126
knn, 70, 72, 127
knn (k nearest neighbours algorithm (k-NN)), 124
knn.cv, 72, 126
knn.cv (Cross-Validation for the k-NN algorithm), 68
kuiper (Uniformity test for circular data), 277
kurt (Skewness and kurtosis coefficients), 258
kurt.test2 (Hypothesis testing between two skewness or kurtosis coefficients), 116
laplace.mle (MLE of continuous univariate distributions defined on the real line), 195
Lbeta, 23, 217
Lbeta (Natural logarithm of the beta function), 218
Lchoose, 217, 246
Lchoose (Binomial coefficient and its logarithm), 22
length.Hash (Hash object), 109
Lgamma, 23, 218
Lgamma (Natural logarithm of the gamma function and its derivatives), 219
Limited number of eigenvalues and eigenvectors of a symmetric matrix, 127
lindley.mle (MLE of continuous univariate distributions defined on the positive line), 193
Linear models for large scale data, 128
list.ftests (Many F-tests with really huge matrices), 143
lm, 129
lm.fit, 129
lmfit, 267
lmfit (Linear models for large scale data), 128
Log, 246
Log (Natural Logarithm each element of a matrix), 217
logcauchy.mle (MLE of continuous univariate distributions defined on the positive line), 193
Logistic and Poisson regression models, 130
Logistic or Poisson regression with a single categorical predictor, 131
logistic.cat1, 13
logistic.cat1 (Logistic or Poisson regression with a single categorical predictor), 131
logistic.mle (MLE of continuous univariate distributions defined on the real line), 195
logistic_only, 20, 21, 63, 102, 107, 126, 131, 132, 143, 162, 167, 174, 179, 211, 235, 251, 281
logistic_only (Many univariate simple logistic and Poisson regressions), 176
logitnorm.mle (MLE of distributions defined in the (0, 1) interval), 198
loglm, 30
loglogistic.mle (MLE of continuous univariate distributions defined on the positive line), 193
lognorm.mle (MLE of continuous univariate distributions defined on the positive line), 193
logseries.mle (MLE of count data (univariate discrete distributions)), 196
lomax.mle (MLE of continuous univariate distributions defined on the positive line), 193
Lower and Upper triangular of a matrix, 133
lower_tri (Lower and Upper triangular of a matrix), 133
Mad (Mean - Median absolute deviation of a vector), 185
mad2 (Mean - Median absolute deviation of a vector), 185
mahala, 77
mahala (Mahalanobis distance), 134
Mahalanobis distance, 134
Many (and one) area under the curve values, 135
Many 2 sample proportions tests, 136
Many 2 sample tests, 137
Many analysis of variance tests with a discrete variable, 139
Many ANCOVAs, 140
Many ANOVAs for count data with Poisson or quasi Poisson models, 141
Many exponential regressions, 142
Many F-tests with really huge matrices, 143
Many G-square and Chi-square tests of independence, 144
Many Gini coefficients, 146
Many hypothesis tests for two means of percentages, 147
Many moment and maximum likelihood estimations of variance components, 148
Many multi-sample tests, 150
Many multivariate simple linear regressions coefficients, 151
Many non parametric multi-sample tests, 152
Many odds ratio tests, 154
Many one sample goodness of fit tests for categorical data, 155
Many one sample tests, 156
Many random intercepts LMMs for balanced data with a single identical covariate, 157

Many regression based tests for single sample repeated measures, 159

Many score based regressions, 161

Many Shapiro-Francia normality tests, 163

Many simple circular or angular regressions, 164

Many simple geometric regressions, 165

Many simple linear mixed model regressions, 166

Many simple linear regressions coefficients, 167

Many simple multinomial regressions, 168

Many simple regressions for positive valued data, 169

Many tests for the dispersion parameter in Poisson distribution, 171

Many two-way ANOVAs, 172

Many univariate generalised linear models, 173

Many univariate simple linear regressions, 175

Many univariate simple logistic and Poisson regressions, 176

Many univariate simple quasi poisson regressions, 178

Many Welch's F-tests, 179

mat.mat (Number of equal columns between two matrices), 221

mat.mult, 228

mat.mult (Matrix multiplication), 181

Match, 60, 87, 117, 180, 221, 271

match, 181

match.coefs (Column-wise matching coefficients), 53

Matrix multiplication, 181

Matrix with all pairs of t-tests, 182

Matrix with G-square tests of independence, 183

matrnorm, 254

matrnorm (Generates random values from a normal and puts them in a matrix), 106

maxboltz.mle (MLE of continuous univariate distributions defined on the positive line), 193

mcnemar (Multi-sample tests for vectors), 209

mcnemars (Many 2 sample tests), 137

Med - Median absolute deviation of a vector, 185

med (Median of a vector), 186

Median, 37, 39, 40, 46, 185, 247

Median (Median of a vector), 186

Median of a vector, 186

mediandir (Spherical and hyperspherical median), 268

min_max (Minimum and maximum), 188

Minima and maxima of two vectors/matrices, 187

Minimum and maximum, 188

Minimum and maximum frequencies, 189

MLE for multivariate discrete data, 190

MLE of (hyper-)spherical distributions, 191

MLE of continuous univariate distributions defined on the positive line, 193

MLE of continuous univariate distributions defined on the real line, 195

MLE of count data (univariate discrete distributions), 196

MLE of distributions defined in the (0, 1) interval, 198

MLE of some circular distributions, 200

MLE of the inverted Dirichlet distribution, 201

MLE of the multivariate (log-) normal distribution, 202

MLE of the multivariate t distribution, 204

MLE of the ordinal model without covariates, 205

MLE of the tobit model, 206

model.matrix, 75

Moment and maximum likelihood estimation of variance components, 207

Multi-sample tests for vectors, 209
multinom.mle, 202, 203
multinom.mle (MLE for multivariate
discrete data), 190
multinom.nb (Naive Bayes classifiers),
215
multinom.reg (Multinomial regression),
211
multinom.regs (Many simple multinomial
regressions), 168
Multinomial regression, 211
multinomnb.pred (Prediction with some
naive Bayes classifiers), 233
Multivariate kurtosis, 212
Multivariate Laplace random values
simulation, 213
Multivariate normal and t random
values simulation, 214
multivmf.mle (MLE of (hyper-)spherical
distributions), 191
mv.eeltest1, 93
mv.eeltest1 (Exponential empirical
likelihood for a one sample
mean vector hypothesis
testing), 90
mv.eeltest2, 91, 124
mv.eeltest2 (Exponential empirical
likelihood hypothesis testing
for two mean vectors), 91
mvbetas, 64, 87, 129, 168, 176, 271
mvbetas (Many multivariate simple
linear regressions
coefficients), 151
mvkurtosis (Multivariate kurtosis), 212
mvlnorm.mle (MLE of the multivariate
(log-) normal distribution),
202
mvlnorm.mle, 74, 204
mvlnorm.mle (MLE of the multivariate
(log-) normal distribution),
202
mvtnorm.mle (MLE of the multivariate t
distribution), 204

Naive Bayes classifiers, 215
Natural Logarithm each element of a
matrix, 217
Natural logarithm of the beta
function, 218
Natural logarithm of the gamma
function and its derivatives, 219
negative (Apply method to Positive and
Negative number), 15
negbin.mle, 172, 191, 275
negbin.mle (MLE of count data
(univariate discrete
distributions)), 196
Norm (Norm of a matrix), 220
Norm of a matrix, 220
normal.mle, 56, 194, 207
normal.mle (MLE of continuous
univariate distributions
defined on the real line), 195
normlog.mle (MLE of continuous
univariate distributions
defined on the positive line),
193
normlog.reg, 105, 120, 170
normlog.reg (Gaussian regression with
a log-link), 105
normlog.regs, 106
normlog.regs (Many simple regressions
for positive valued data), 169
nth, 10, 16, 25, 43, 55, 57, 76, 108, 117, 133,
186, 188, 189, 244, 247–249, 261,
264, 277
nth (Column and row-wise nth smallest
value of a matrix/vector), 39
Number of equal columns between two
matrices, 221

odds, 54, 59, 223
odds (Many odds ratio tests), 154
Odds ratio and relative risk, 222
odds.ratio, 154, 282
odds.ratio (Odds ratio and relative
risk), 222
omp (Orthogonal matching pursuit
variable selection), 226
ompr, 95
ompr (Orthogonal matching pursuit
variable selection), 226
One sample t-test for a vector, 223
Operations between two matrices or
matrix and vector, 224
Order, 247
Order (Column and row-wise Order - Sort Indices), 41
ordinal.mle (MLE of the ordinal model without covariates), 205
Orthogonal matching pursuit variable selection, 226
Outer (Outer function), 227
Outer function, 227
pareto.mle (MLE of continuous univariate distributions defined on the positive line), 193
pc.skel, 230, 276
pc.skel (Skeleton of the PC algorithm), 256
percent.ttest (Hypothesis test for two means of percentages), 114
percent.ttests (Many hypothesis tests for two means of percentages), 147
permcor (Permutation based p-value for the Pearson correlation coefficient), 229
Permutation, 228
permutation, 60
permutation (Permutation), 228
Permutation based p-value for the Pearson correlation coefficient, 229
Pmax (Minima and maxima of two vectors/matrices), 187
Pmin (Minima and maxima of two vectors/matrices), 187
Pmin_Pmax (Minima and maxima of two vectors/matrices), 187
pois.test (Tests for the dispersion parameter in Poisson distribution), 274
poisdisp.test (Tests for the dispersion parameter in Poisson distribution), 274
poisson.anova, 98, 132, 139, 142, 172, 275
poisson.anova (Analysis of variance with a count variable), 12
poisson.anovas, 13, 132, 172, 275
poisson.anovas (Many analysis of variance tests with a discrete variable), 139
poisson.cat1 (Logistic or Poisson regression with a single categorical predictor), 131
poisson.mle, 13, 56, 98, 139, 172, 191, 275
poisson.mle (MLE of count data (univariate discrete distributions)), 196
poisson.nb, 191
poisson.nb (Naive Bayes classifiers), 215
poisson_only, 13, 20, 21, 63, 98, 102, 131, 132, 139, 143, 162, 166, 169, 172, 174, 179, 198, 237, 275, 281
poisson_only (Many univariate simple logistic and Poisson regressions), 176
poissonnb.pred (Prediction with some naive Bayes classifiers), 233
poly.cor (Polyserial correlation), 230
Polyserial correlation, 230
Pooled covariance matrix, 232
pooled.cov (Pooled covariance matrix), 232
positive (Apply method to Positive and Negative number), 15
Prediction with some naive Bayes classifiers, 233
print.Hash (Hash object), 109
print.iterator (Iterator), 121
print.ufactor (Fast and general - untyped representation of a factor variable), 93
prop.reg, 114, 147, 148, 237, 251
prop.reg (Quasi binomial regression for proportions), 234
prop.regs, 106, 166, 169, 170, 177
prop.regs (Quasi binomial regression for proportions), 234
proptest (Many one sample tests), 156
proptests (Many 2 sample proportions tests), 136
qpois.reg, 15, 251
qpois.reg (Quasi Poisson regression for count data), 236
qpois.regs (Quasi Poisson regression for count data), 236
Quasi binomial regression for proportions, 234
Quasi Poisson regression for count data, 236
quasi.poisson_only, 177
quasi.poisson_only (Many univariate simple quasi poisson regressions), 178
quasipoisson.anova, 15
quasipoisson.anova (Analysis of variance with a count variable), 12
quasipoisson.anovas (Many analysis of variance tests with a discrete variable), 139
racg, 192, 213, 215
racg (Angular central Gaussian random values simulation), 13
Random intercepts linear mixed models, 237
Random values simulation from a von Mises distribution, 239
Rank, 45
Rank (Ranks of the values of a vector), 240
Ranks of the values of a vector, 240
rayleigh.mle (MLE of continuous univariate distributions defined on the positive line), 193
rbing, 253
rbing (Simulation of random values from a Bingham distribution), 252
rbingham (Simulation of random values from a Bingham distribution with any symmetric matrix), 253
read.directory, 26, 28, 72, 118, 122, 245, 265, 273
read.directory (Reading the files of a directory), 241
read.examples, 28
read.examples (Reading the files of a directory), 241
Reading the files of a directory, 241
regression, 20, 21, 63, 129, 131, 143, 162, 174, 177, 179, 180, 281
regression (Many univariate simple linear regressions), 175
rel.risk (Odds ratio and relative risk), 222
rep_col (Replicate columns/rows), 243
rep_row (Replicate columns/rows), 243
Repeated measures anova, 242
Replicate columns/rows, 243
Represantation of Stack, 244
Rfast-package, 6
rint.mle (Moment and maximum likelihood estimation of variance components), 207
rint.reg, 158, 160, 167, 208, 243
rint.reg (Random intercepts linear mixed models), 237
rint.regbx, 158, 160, 208
rint.regbxs (Random intercepts linear mixed models), 237
rint regs (Many simple linear mixed model regressions), 166
rm.anova (Repeated measures anova), 242
rm.anovas, 88, 243
rm.anovas (Many regression based tests for single sample repeated measures), 159
rm.lines, 88, 158, 239
rm.lines (Many regression based tests for single sample repeated measures), 159
rmdp, 82
rmdp (High dimensional MCD based detection of outliers), 111
rmvlaplace, 14, 215
rmvlaplace (Multivariate Laplace random values simulation), 213
rmvnorm, 14, 74, 107, 213, 254
rmvnorm (Multivariate normal and t random values simulation), 214
rmvt, 14, 74, 213, 215
rmvt (Multivariate normal and t random values simulation), 214
Rnorm, 107
Rnorm (Simulation of random values from a normal distribution), 254
Round, 16, 261
Round (Round each element of a matrix/vector), 245
Round each element of a matrix/vector, 245
Row - Wise matrix/vector count the frequency of a value, 246
Row-wise minimum and maximum, 247
Row-wise true value, 248
rowAll (Column and row-wise Any/All), 36
rowAny (Column and row-wise Any/All), 36
rowCountValues (Row - Wise matrix/vector count the frequency of a value), 246
rowcvs (Column and row wise coefficients of variation), 35
rowFalse, 25, 57, 108, 244
rowFalse (Row-wise true value), 248
rowhameans (Column and row-wise means of a matrix), 37
rowMads, 225
rowMads (Column and rows-wise mean absolute deviations), 49
rowMaxs, 43, 55, 188, 189
rowMaxs (Row-wise minimum and maximum), 247
rowMinsMaxs (Row-wise minimum and maximum), 247
rowMins (Row-wise minimum and maximum), 247
rowMedians (Column and row-wise median), 37
rowMedians (Column and row-wise medians), 38
rowMins, 25, 43, 55, 57, 76, 108, 133, 188, 189, 244, 249
rowMins (Row-wise minimum and maximum), 247
rowSort (Sorting of the columns-rows of a matrix), 263
rowsums, 38
rowsums (Column and row-wise sums of a matrix), 46
rowTabulate (Column and row-wise tabulate), 47
rowTrue, 25, 57, 108, 244
rowTrue (Row-wise true value), 248
rowTrueFalse (Row-wise true value), 248
rowVars, 76, 133, 249
rowVars (Column and row-wise variances and standard deviations), 48
rvmf, 107, 192, 201, 240, 252, 254
rvmf (Simulation of random values from a von Mises-Fisher distribution), 255
rvonmises, 58, 107, 201, 254, 256, 278
rvonmises (Random values simulation from a von Mises distribution), 239
score.betaregs (Many score based regressions), 161
score.expregs (Many score based regressions), 161
score.gammaregs (Many score based regressions), 161
score.geomregs, 166, 169
score.geomregs (Many score based regressions), 161
score.invgaussregs (Many score based regressions), 161
score.multinomregs, 211
score.multinomregs (Many score based regressions), 161
score.negbinregs (Many score based regressions), 161
score.weibregs (Many score based regressions), 161
score.ztpregs (Many score based regressions), 161
Search for variables with zero range in a matrix, 249
INDEX

sftest (Many Shapiro-Francia normality tests), 163
sftests, 52, 107
sftests (Many Shapiro-Francia normality tests), 163
Significance testing for the coefficients of Quasi binomial or the quasi Poisson regression, 250
Simulation of random values from a Bingham distribution, 252
Simulation of random values from a Bingham distribution with any symmetric matrix, 253
Simulation of random values from a normal distribution, 254
Simulation of random values from a von Mises-Fisher distribution, 255
Skeleton of the PC algorithm, 256
Skew, 52, 116
skew (Skewness and kurtosis coefficients), 258
skew.test2, 52, 212, 259
skew.test2 (Hypothesis testing between two skewness or kurtosis coefficients), 116
Skewness and kurtosis coefficients, 258
Some summary statistics of a vector for each level of a grouping variable, 259
Sort, 50, 187
Sort (Sort - Integer Sort - Sort a vector corresponding to another), 260
Sort - Integer Sort - Sort a vector corresponding to another, 260
Sort and unique numbers, 262
sort_cor_vectors, 263, 264
sort_cor_vectors (Sort - Integer Sort - Sort a vector corresponding to another), 260
sort_mat (Sorting of the columns-rows of a matrix), 263
sort_unique, 16, 261, 264
sort_unique (Sort and unique numbers), 262
Sorting of the columns-rows of a matrix, 263
Source many R files, 264
sourceR, 28, 242
sourceR (Source many R files), 264
sourceRd, 28, 242
sourceRd (Source many R files), 264
spat.med, 232, 267, 268
spat.med (Spatial median for Euclidean data), 265
Spatial median for Euclidean data, 265
Spatial median regression, 266
Spatial sign covariance matrix, 267
spatmed.reg, 232, 268
spatmed.reg (Spatial median regression), 266
spdinv (Inverse of a symmetric positive definite matrix), 120
Spherical and hyperspherical median, 268
spml.mle, 33, 127, 165
spml.mle (MLE of some circular distributions), 200
spml.reg, 34
spml.reg (Circular or angular regression), 32
spml regs (Many simple circular or angular regressions), 164
squareform (Vector allocation in a symmetric matrix), 279
sscov, 267
sscov (Spatial sign covariance matrix), 267
Standardisation, 269
standardise (Standardisation), 269
Sub-matrix, 270
submatrix (Sub-matrix), 270
Sum of all pairwise distances in a distance matrix, 271
Table, 68, 231
Table (Table Creation - Frequency of each value), 272
Table Creation - Frequency of each value, 272
Tcrossprod, 66, 81
Tcrossprod (Matrix multiplication), 181
Tests for the dispersion parameter in Poisson distribution, 274
tmle (MLE of continuous univariate distributions defined on the
real line), 195
tobit.mle (MLE of the tobit model), 206
Topological sort of a DAG, 275
topological_sort (Topological sort of a DAG), 275
total.dist, 77, 86
total.dist (Sum of all pairwise distances in a distance matrix), 271
total.dista, 77, 86
total.dista (Sum of all pairwise distances in a distance matrix), 271
transpose, 181
transpose (Transpose of a matrix), 276
Transpose of a matrix, 276
Trigamma (Natural logarithm of the gamma function and its derivatives), 219
test, 84, 136, 138, 155, 164, 183, 224
test (Many one sample tests), 156
test1, 83
test1 (One sample t-test for a vector), 223
test2, 24, 90, 114, 142, 148
test2 (Multi-sample tests for vectors), 209
tests, 12, 84, 136, 137, 141, 144, 151, 155, 157, 164, 173, 183, 210, 224
tests (Many 2 sample tests), 137
tests.pairs (Matrix with all pairs of t-tests), 182
twoway.anova (Multi-sample tests for vectors), 209
twoway.anovas (Many two-way ANOVAs), 172
ufactor (Fast and general - untyped representation of a factor variable), 93
Uniformity test for circular data, 277
univglm (Many univariate generalised linear models), 173
univglm (Many univariate generalised linear models), 173
upper_tri (Lower and Upper triangular of a matrix), 133
Var (Variance of a vector), 278
var2test (Multi-sample tests for vectors), 209
var2tests (Many 2 sample tests), 137
varcomps.mle, 88, 149, 160, 243
varcomps.mle (Moment and maximum likelihood estimation of variance components), 207
varcomps.mom, 158, 239
varcomps.mom (Moment and maximum likelihood estimation of variance components), 207
Variance of a vector, 278
vartest (Many one sample tests), 156
vartests (Many multi-sample tests), 150
vecdist, 228
vecdist (Distance matrix), 78
Vector allocation in a symmetric matrix, 279
vm.mle, 56, 192, 196, 240
vm.mle (MLE of some circular distributions), 200
vmf.mle, 58, 115, 127, 201, 256, 269, 278
vmf.mle (MLE of (hyper-)spherical distributions), 191
watson, 58
watson (Uniformity test for circular data), 277
weib.reg (Weibull regression model), 280
Weibull regression model, 280
weibull.mle (MLE of continuous univariate distributions defined on the positive line), 193
which.is (Index of the columns of a data.frame which are a specific type), 117
wigner.mle (MLE of continuous univariate distributions defined on the real line), 195
wrapcauchy.mle (MLE of some circular distributions), 200
XopY.sum (Operations between two matrices or matrix and vector), 224
yule, 59
yule (Yule's Y (coefficient of colligation)), 281
Yule's Y (coefficient of colligation), 281

zip.mle, 191, 194, 196
zip.mle (MLE of count data (univariate discrete distributions)), 196
ztp.mle, 191
ztp.mle (MLE of count data (univariate discrete distributions)), 196